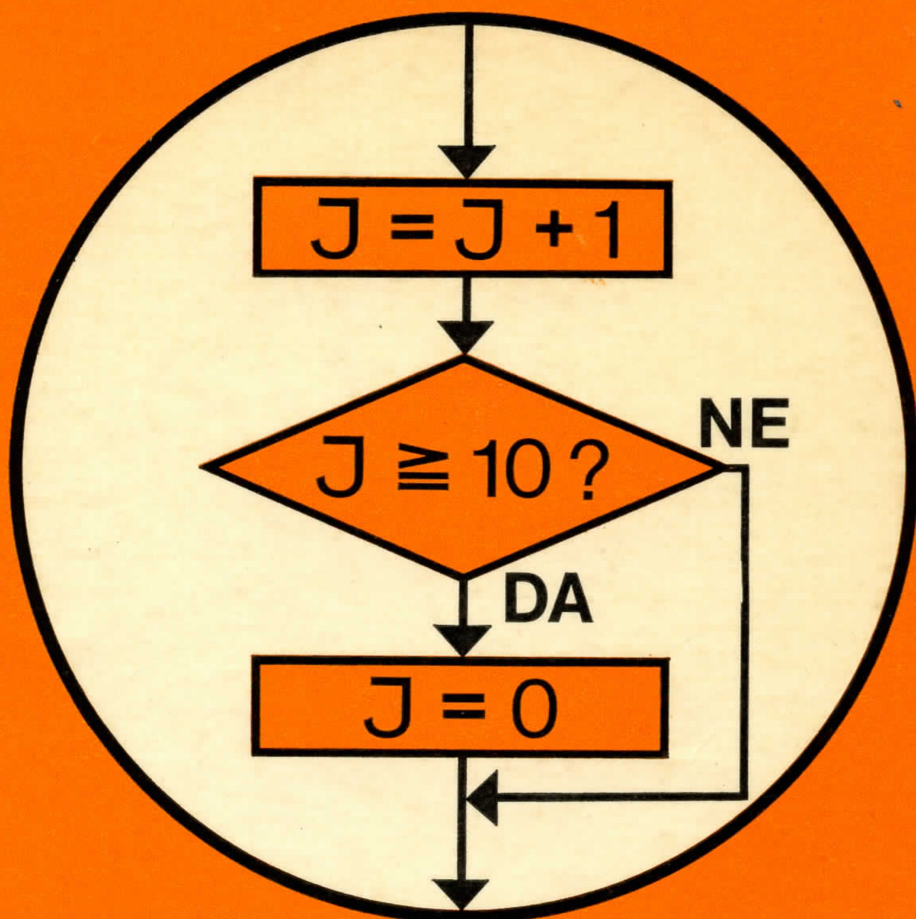


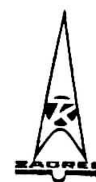
Božidar Stefanini

FORTRAN
Udžbenik programiranja

Božidar Stefanini

F O R T R A N

udžbenik programiranja



Dr ing. BOŽIDAR STEFANINI
Redovni profesor Elektrotehničkog fakulteta Sveučilišta u Zagrebu

FORTRAN

UDŽBENIK PROGRAMIRANJA

TEHNIČKA KNJIGA
ZAGREB

PREDGOVOR

Ova knjiga namijenjena je onima koji počinju učiti programiranje za elektronička računala, a ograničava se samo na jedan simbolički jezik, i to FORTRAN. To je, doduše, jezik pomoću kojega čovjek komunicira sa strojem, ali je ipak sličan jezicima koji služe za komuniciranje među ljudima. Zbog toga je knjiga pisana na način na koji bi bio pisan udžbenik bilo kojeg jezika. U svakoj lekciji dan je po jedan program u FORTRAN-u, koji služi kao uzorno štivo, a osim toga su dana posebna objašnjenja koja služe kao sintaktička i ostala pravila.

Knjiga je nastala na osnovi predavanja i skripata u okviru tečaja za programiranje, koje su organizirali Elektrotehničko društvo Zagreb i Elektrotehnički fakultet u Zagrebu. Pri pisanju ove knjige iskorištena su iskustva stečena prilikom održavanja tih tečajeva.

Ovaj jezik ima više varijanata, ali je u knjizi obrađena samo jedna, FORTRAN IV za računala IBM 1130. Na taj način ovaj udžbenik postaje posve konkretan, za učenje takvog jezika koji može biti odmah direktno primijenjen, što je naročito važno kod praktičkih vježbi na računalu.

Tko nauči FORTRAN iz ove knjige, moći će se vrlo dobro služiti elektroničkim računalom IBM 1130. Ali će ujedno steći i općenito znanje o koncepciji i konstrukciji simboličkih jezika. Tako će mu biti veoma lako dopuniti svoje znanje za FORTRAN IV u cijelosti, a isto tako naučiti koju drugu varijantu FORTRAN-a. Neće imati poteškoća ni da nauči neki drugi simbolički jezik, pogotovu ako je taj (kao i FORTRAN) prvenstveno namijenjen rješavanju matematičkih i tehničkih problema, npr. ALGOL.

Knjiga je pisana u tri poglavlja, uz pretpostavku da se nakon poglavlja izradi više programa, i ti programi izvrše na elektroničkom računalu. Pri tome se prvenstveno misli na računala IBM 1130. Ta računala su većinom opremljena MONITOR-om, koji upravlja radom i uvelike rasterećuje operatera. Zbog toga ova knjiga obuhvaća i najnužnije o korištenju MONITOR-a, bar ono što je neophodno pri pisanju programa za direktno korištenje na računalu.

Knjiga je pisana na principu koji pokazuje kako treba ispravno pisati program, koji onda može poslužiti kao uzor u svakom pogledu: kako riješiti matematički problem, kako organizirati program, kako međusobno uskladiti pojedine naredbe u programu i konačno, kako ispravno pisati te naredbe. S naročitom namjerom nije pokazano kako ne valja raditi, osim kad na pogrešku računalo reagira na posve specifičan način.

U svakoj lekciji na početku nalazi se program (pisan u FORTRAN-u), u kojem se pomalo i logično uvodi nova materija. Svaki je program potpuna cjelina, a ne samo neki isječak, pa će računalo takav program bez daljnjega izvršiti. To je učinjeno zato jer je iskustvo pokazalo da uvođenje materije pomoću primjera na isječcima programa ostavlja dojam nedorečenog i razvija nesigurnost kod onoga koji to uči.

Pri pisanju knjige je usvojen princip da čim prije treba početi praktički raditi na računalu. Bez praktikuma na računalu ne može se naučiti programiranje, isto kao što se bez sviranja ne može naučiti klavir. Zbog toga je materija podijeljena u tri poglavlja, na taj način da se već nakon prvog poglavlja može pristupiti praktikumu. Da se praktikum olakša, iza svakog poglavlja naveden je niz zadataka koji se mogu riješiti uz poznavanje do tada usvojenog znanja.

Knjiga zbog toga može veoma dobro poslužiti kod tečajeva, u okviru kojih je organiziran i praktikum. No ona se može koristiti i pojedinačno izvan tečajeva, ali se svima preporučuje da stečeno znanje provjere praktički na računalu. Iza svakog poglavlja treba riješiti bar nekoliko raznolikih zadataka, i napisane programe za računalo izvršiti.

Materijal u ovoj knjizi je pisan sažeto, pa ga treba studirati, a ne samo čitati. Pri tome treba uspoređivati napisani tekst s pripadnim programom, shemom toka, ulaznim podacima i izlaznim rezultatima. Na slijedeću lekciju treba prijeći tek onda kada smo prethodnu dobro proučili, a kasnije po potrebi ponoviti po koji dio iz prijašnjih lekcija.

Ova knjiga bila je napisana u vrijeme kad je od fakulteta u Zagrebu jedino Elektrotehnički fakultet imao elektroničko računalo, i to IBM 1130. Razumljivo je da je materija prilagođena tom računalu, na kojem se vršila redovna nastava i različiti tečajevi.

Dok IBM 1130 upotrebljava osnovni FORTRAN IV, dotle UNIVAC serije 1100 upotrebljava FORTRAN V, koji je mnogo opširniji, ali se u osnovnom dijelu bitno ne razlikuje od FORTRAN-a IV. Ova knjiga može se dakle vrlo dobro koristiti i za učenje FORTRAN-a V (za UNIVAC seriju 1100), te predstavlja osnovni dio tog jezika. Ipak postoje neke male razlike, koje u biti nisu tipične za FORTRAN nego za računalo s kojim se radi. Te su razlike navedene u dodatku na kraju knjige.

Na ovome mjestu želim zahvaliti mr. Alfredu Žepiću, koji mi je pomogao u sastavljanju zadataka, kao i svima onima koji su direktno ili indirektno doprinijeli da knjiga iziđe u ovom obliku. Bit ću zahvalan svakome tko mi svojim sugestijama i primjedbama ukaže što bi se u knjizi moglo poboljšati.

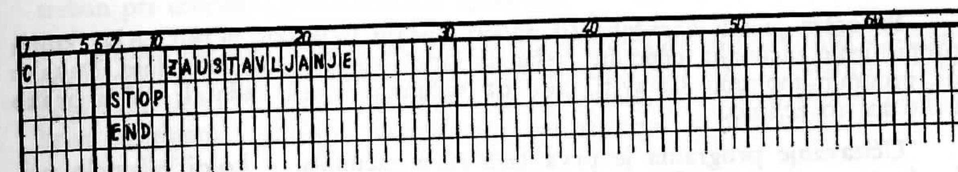
Nadam se da će ova knjiga uspješno doprinijeti što širem i što boljem korištenju elektroničkih računala u nas.

B. S.

Sadržaj

LEKCIJA **1.**

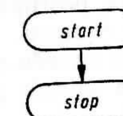
POČETNI POJMOVI



Evo jednog potpunog programa u FORTRAN-u, pa i više od toga, jer ima i komentar na našem jeziku. Ovaj program elektroničko računalo bi prihvatilo i izvršilo. Na njemu ćemo već u početku mnogo toga naučiti. I u slijedećim lekcijama učit ćemo na primjerima potpunih programa, koji se — onakvi kakvi su ovdje napisani — mogu bez daljnjega dati računalu na izvršenje.

Zadatak koji pomoću ovog prvog programa dajemo računalu veoma je jednostavan: čim računalo krene, treba da se zaustavi. To će računalo i učiniti kad budu izvršilo ovaj program. Dakako, programi u praksi nisu jednostavni, i od izvršenja ovog programa ne bismo u stvari imali nikakve koristi. Ali za učenje programiranja ovakav jednostavni program veoma je koristan, pa će i u slijedećim lekcijama programi biti što jednostavniji.

Kad je zadatak koji želimo riješiti pomoću elektroničkog računala definiran, moramo postaviti postupak koji vodi do rješenja. Taj postupak mora točno definirati svaki korak od početka pa do kraja izvršenja zadatka. To najradije prikazujemo grafički pomoću sheme toka, prema kojoj je onda lako napisati program. U ovoj lekciji zadatak je veoma jednostavan, pa je i shema toka jednostavna ali, kao uvijek, i korisna.



Rezultat izvršenja ovog programa također je veoma jednostavan. Čim računalo pokrenemo, ono će se zaustaviti. I to je sve. A sada razmotrimo na osnovi ovog primjera neke početne pojmove.

FORTRAN je simbolički jezik za pisanje programa. Jednostavan je za čovjeka, ali je točno definiran, pa ga i računalo može prihvatiti. Postoje mnogi simbolički jezici za elektronička računala (FORTRAN, ALGOL, COBOL itd.),

no mi ćemo se ovdje baviti samo FORTRAN-om, kao što je u uvodu rečeno. Da bi računalo moglo »razumjeti« neki simbolički jezik, mora posjedovati odgovarajući kompilator za prevođenje programa od simboličkog jezika na svoj interni jezik stroja. No o svemu tome se brine operater kod računala. Korisnik treba samo da napiše program u simboličkom jeziku, u našem slučaju u FORTRAN-u.

FORTRAN-formular je formular za pisanje programa u FORTRAN-u (vidi prilog 1). Formular ima 80 stupaca, podijeljenih na polja, u koja se na određeni način upisuju pojedine naredbe programa. Svaka naredba upisuje se u poseban redak. To će poslije operateri prenijeti na bušene kartice (svaki redak na jednu karticu) i predati računalu.

Program je niz naredaba (u našem slučaju u jeziku FORTRAN) kojim propisujemo računalu točno što treba da radi, i to točno kojim redom. Dakako da program moramo sastaviti tako da računalo izvrši zadatak točno onako kako mi to želimo.

Učitavanje programa je prva faza rada računala, u kojoj računalo prihvaća program, prevodi ga na svoj interni jezik i pamti ga. Sad je računalo spremno da taj program izvede koliko god puta to želimo.

Izvršenje programa je druga faza rada računala, u kojoj računalo izvršava zapamćeni program. Razbijanje rada računala na dvije faze omogućuje da se program napiše posve općenito i izvrši proizvoljno puta, svaki put s različitim brojčanim podacima. To će posebice doći do izražaja u drugom i trećem poglavlju.

Naredba je osnovni element programa. Niz naredaba, koje se nižu jedna za drugom po određenom redoslijedu, sačinjava program. Naredba može biti organizaciona ili izvršna.

Organizaciona naredba koristi se samo u prvoj fazi, tj. za vrijeme učitavanja. Prema njima računalo organizira svoj rad. Organizaciona naredba ne izvršava se za vrijeme izvršenja programa.

Izvršna naredba je takva koju računalo za vrijeme učitavanja programa samo zapamti, ali je u drugoj fazi, tj. pri izvršenju programa izvrši, i to onda kad na nju dođe red.

Shema toka je grafički prikaz postupka prema kojem želimo napisati program. Shemu toka treba uvijek nacrtati, pa ma kako postupak bio jednostavan. Ona nam daje vizuelnu vezu između postavljenog postupka i napisanog programa, te nam znatno olakšava pisanje programa i izbjegavanje logičkih pogrešaka.

Oval u shemi toka se upotrebljava za start, stop i slične naredbe.

Start u shemi toka označava početak izvršenja programa. Za njega u FORTRAN-u ne postoji odgovarajuća naredba, jer se pokretanje računala mora izvršiti ručno (to obavlja operater pritiskom na odgovarajuću tipku).

STOP je izvršna FORTRAN-naredba, koja znači da prilikom izvršenja programa, kad ta naredba dođe na red, računalo treba da se zaustavi.

END je organiziciona FORTRAN-naredba, koja označuje završetak programa. Program mora uvijek završiti tom naredbom. Kad računalo prilikom učitavanja programa (prva faza!) dođe do te naredbe, onda smatra program završenim i ne učitava dalje.

C (u prvoj koloni!) je organizaciona naredba, koja »kaže« računalu da je ono što slijedi iza toga samo komentar za udobnost čovjeku. Računalo se s tim dalje ne bavi.

Centralna jedinica je osnovni dio računala s memorijom, aritmetičkim dijelom i organizacionim dijelom. Za izvršenje ovog programa dovoljna je samo centralna jedinica, i to memorija i organizacioni dio. Aritmetički dio bit će potreban pri izvršenju matematičkih operacija.

Čitalo kartica je jedan od perifernih jedinica računala i služi za čitanje podataka s bušenih kartica. U ovom programu (a isto tako u ostalim u prvom poglavlju) čitalo kartica služi samo za učitavanja programa. To je međutim briga operatera, i korisnik računala o tome ne mora voditi računa.

2. LEKCIJA

PISANJE i TISKANJE

```
C     STAMPANJE TEKSTA
      WRITE(3,100)
100   FORMAT(48H ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789.,'=+-*/())
      STOP
      END
```

Zadatak ovog programa jest da računalo tiska sve raspoložive znakove, tj. slova, brojke i specijalne znakove. Shema toka prikazuje postupak za izvršenje tog zadatka, koji je još uvijek veoma jednostavan.

start

A...$ (100)

stop

Ako taj program damo računalu na izvršenje, dobit ćemo ovaj rezultat:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789.,'=+-*/()
```

Kao i prije, opet ćemo razmotriti pojmove koje možemo naučiti iz ovog programa.

WRITE (......., ...) je izvršna naredba za izlaz informacije, tj. za izdavanje rezultata rada računala. U zagradi se na prvom mjestu definira izlazna jedinica (gdje se izdaju informacije), a na drugom mjestu format (u kakvoj formi se izdaju informacije).

3 (na prvom mjestu u zagradi iza WRITE) određuje da se izlazne informacije moraju dati na linijskoj tiskaljci. Brojka 3 je karakteristika izlazne jedinice, i konkretno linijske tiskaljke. Druge brojke karakteriziraju druge izlazne jedinice, no mi ćemo ovdje koristiti samo linijsku tiskaljku.

Linijska tiskaljka je brza izlazna jedinica te tiska odjednom čitav redak sa 120 znakova, brzinom od 80 redaka u minuti. Na jednu stranicu stanu 72 retka (pri gustoći od 6 redaka po colu), odnosno 96 redaka (pri gustoći od 8 redaka po colu).

Broj formata (na drugom mjestu u zagradi iza WRITE) može biti bilo koji broj do pet znamenaka, a pokazuje na koji se format ta naredba odnosi. Isti broj mora se nalaziti ispred naredbe FORMAT, koja je vezana na tu naredbu WRITE.

FORMAT je organizaciona naredba, koja određuje u kojoj će se formi izdati informacije. To je specificirano u zagradi iza riječi FORMAT. U ovom programu korišten je samo jedan način izdavanja informacija, daljnji načini bit će korišteni kasnije. Naredba FORMAT mora imati svoj broj, preko kojeg je vezana s pripadnom naredbom WRITE.

H-format koristi se za tiskanje teksta. U zgradi iza FORMAT napiše se najprije koliko znakova treba doslovna tiskati. Treba paziti da se broj ispred H slaže s dužinom teksta.

Prvi znak u retku neće biti otiskan, ako izlazne informacije dajemo putem linijske tiskaljke. Zato ćemo do daljnjega umjesto prvog znaka u retku ostavljati bjeline. Te bjeline u rezultatu tiskanja nema.

Znakovi koje možemo upotrijebiti navedeni su u samom programu i u rezultatu tiskanja. To su 26 slova engleske abecede, 10 brojčanih znamenaka, 10 posebnih znakova i bjelina.

Trapez (odozdo širi) upotrebljava se u shemi toka kao simbol za izlaz informacija (u našem slučaju tiskanje).

Izlazna lista je papir s otisnutim izlaznim informacijama, tj. s otisnutim rezultatima rada računala.

Pisanje znakova u formularu mora biti jasno i jednoznačno. Svaki znak neka bude unutar svoje kućice (radije pisan uspravno, nego koso) i pisan prema dogovorenom obliku (mi ćemo upotrebljavati način pisanja kao u ovom programu).

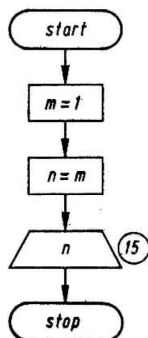
Bjelina se u formularu označuje tako da se kućica ostavi prazna (implicitno označavanje bjeline).

3. LEKCIJA

BROJČANA VRIJEDNOST VARIJABLE

```
C     PRIDAVANJE BROJCANE VRIJEDNOSTI
      M=1
      N=M
C     STAMPANJE BROJCANE VRIJEDNOSTI
      WRITE(3,15)N
   15 FORMAT(I5)
      STOP
      END
```

Zadatak u ovoj lekciji sastoji se u tome da se jednoj varijabli (nazovimo je *m*) prida određena brojčana vrijednost (konkretno vrijednost 1) i da se drugoj varijabli (nazovimo je *n*) prida brojčana vrijednost prve varijable. Konačno treba otisnuti vrijednost druge varijable. Postupak za provedbu ovog zadatka je posve jednostavan i ne treba ga posebno opisivati. Shema toka u kojoj su upotrijebljeni simboli u formulama, slično kao što je uobičajeno u matematici, pokazuje potrebni postupak.



Rezultat rada računala bit će slijedećeg oblika:

1

Brojčano računanje je tipično za rad elektroničkog računala, jer ono ne operira s općim izrazima, već s određenim brojčanim vrijednostima.

= je izvršna naredba za pridavanje brojčane vrijednosti. Varijabli s lijeve strane znaka = pridaje se brojčana vrijednost onoga što se nalazi s desne strane znaka =. Na primjer, u našem slučaju: »neka *m* bude jednako 1«.

Aritmetička naredba obuhvaća varijablu kojoj se pridaje određena brojčana vrijednost, znak jednakosti i izraz s desne strane, čija se brojčana vrijednost pridaje varijabli. (U ovoj lekciji izrazi s desne strane su veoma jednostavni). Iako ima oblik jednadžbe (npr. $n = m$), ipak to nije jednadžba već naredba (u ovom primjeru: »neka *n* bude jednako brojčanoj vrijednosti od *m*«).

Varijabla kojoj se pridaje brojčana vrijednost mora stajati s lijeve strane znaka jednakosti, i to sama bez ičega drugog (predznak, koeficijent ili slično).

Izraz s desne strane znaka jednakosti mora imati već otprije utvrđenu brojčanu vrijednost za sve varijable koje se u njemu nalaze. Na primjer, dvije aritmetičke naredbe u ovom programu ne bi smjele imati obrnut redoslijed.

WRITE (......., ...) **N** je izvršna naredba za izdavanje brojčane vrijednosti varijable *n*.

I-format specificira tiskanje brojčane vrijednosti cjelobrojne varijable. U zagradi iza FORMAT mora se nalaziti slovo I, a iza njega cio broj, koji određuje širinu polja unutar kojeg će se ta brojčana vrijednost otisnuti (u ovom slučaju 5 kućica). Rezultat se otisne pozicioniran udesno.

Pravokutnik u shemi toka upotrebljava se za aritmetičke naredbe.

4. LEKCIJA

CJELOBROJNE VARIJABLE

```
C           IMENA CJELOBROJNIH VARIJABLA
      I=1
      JANKO=2
      K2R=3
      LLL=4
      MAX1=5
      N1234=6
C           JEDNOSTRUKI FORMAT
      WRITE(3,12)I,JANKO,K2R,LLL,MAX1,N1234
12    FORMAT(I3,I4,I5,I6,I7,I8)
C           VISESTRUKI FORMAT
      WRITE(3,13)I,JANKO,K2R,LLL,MAX1,N1234
   13 FORMAT(6I5)
      STOP
      END
```


Zadatak je ovdje da se određenom broju varijabli prida neka brojčana vrijednost, i to uvijek vrijednost cijelog broja. Onda i takve varijable nazivamo cjelobrojne. Poslije treba brojčane vrijednosti tih varijabla otisnuti, i to na dva različita načina. Svrha je ove lekcije da naučimo kako se sastavljaju imena za cjelobrojne varijable, a ujedno i daljnje mogućnosti kod formata. Shema toka prikazuje postupak.

Rezultat rada računala (izlazna lista) izgleda ovako:

```
  1   2    3     4      5       6
    1    2    3    4    5    6
```


Varijabla je veličina kojoj tokom izvršenja pograma mogu biti pridane različite brojčane vrijednosti.

Cjelobrojna varijabla je varijabla čija brojčana vrijednost može biti samo cio broj. Cjelobrojne varijable upotrebljavaju se za računanje s cijelim brojevima, za indekse, za brojenje unutar programa i slično.

Ime cjelobrojne varijable mora početi slovom, i to slovom I, J, K, L, M ili N. Time što je prvo slovo jedno od navedenih, implicitno je specificirano da je varijabla cjelobrojna. Ime može ukupno imati najviše 5 znakova, i to samo slova i brojaka. Posebni znakovi ne mogu se upotrijebiti u imenu varijable.

WRITE (......, ...) I, J, K je izvršna naredba za tiskanje brojčane vrijednosti od većeg broja varijabli (u ovom slučaju su to tri varijable: I, J i K). Broj varijabli u istoj naredbi nije ograničen. Imena varijabla moraju biti odvojena zarezom.

Jednostruki format specificira način tiskanja za jednu varijablu. Više jednostrukih formata mogu biti nanizani jedan za drugim, i u tom slučaju odvajaju se zarezom.

Višestruki format specificira isti način tiskanja za veći broj varijabla. Na koliko se varijabla taj format odnosi, određuje broj koji se nalazi ispred slova I (isto će vrijediti i za druge formate, a ne samo za I-format).

Novi redak kod tiskanja dobiva se pri izvršenju programa uvijek kad dođe na red izvršenje naredbe WRITE (koja se poziva na određeni FORMAT), jer je organizaciona naredba za novi redak implicirana u naredbi FORMAT.

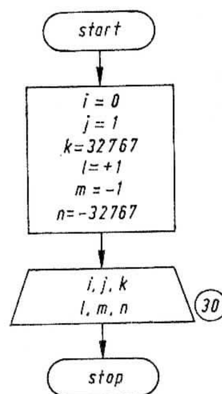
Podudaranje po broju i redoslijedu mora postojati između varijabli u naredbi WRITE i specifikacija u naredbi FORMAT. Isto podudaranje postoji dakako i s brojčanim rezultatim otisnutim u izlaznoj listi.

5. LEKCIJA

CJELOBROJNE KONSTANTE

```
C          PISANJE CJELOBROJNIH KONSTANATA
      I=0
      J=1
      K=32767
C          PISANJE PREDZNAKA
      L=+1
      M=-1
      N=-32767
C          SIRINA FORMATA ZA STAMPANJE
      WRITE(3,30)I,J,K,L,M,N
   30 FORMAT(6I5)
      STOP
      END
```

Zadatak ove lekcije je opet da cjelobrojnim varijablama pridamo cjelobrojne brojčane vrijednosti, ali ovaj puta zato da naučimo ono što je potrebno o pisanju konstanata u programu. Rezultate ćemo opet tiskati, pri čemu dolazi do izražaja širina formata. Postupak je opet prikazan na shemi toka.



Izlazna lista s otisnutim rezultatima izgleda ovako:

```
    0    132767    1   -1*****
```

Konstanta je brojčana vrijednost napisana unutar programa.

Cjelobrojne konstante u programu pišu se onako kako je i inače uobičajeno.

Najveća cjelobrojna konstanta iznosi 32767. Veća vrijednost od ove ne smije se upotrijebiti u programu.

Najmanja cjelobrojna konstanta iznosi — 32767. Manja vrijednost od ove ne smije se upotrijebiti u programu.

Pozitivni predznak kod konstanata u programu ne treba pisati, te se implicitno podrazumijeva. Ipak je dopušteno da se pozitivni predznak napiše (eksplicitno).

Negativni predznak kod konstanata u programu mora se pisati.

Rezulat je izlazni brojčani podatak otisnut na izlaznoj listi.

Pozitivni rezultat otisne se na izlaznoj listi bez predznaka, te se implicitno podrazumijeva.

Negativni rezultat tiska se na izlaznoj listi sa predznakom minus.

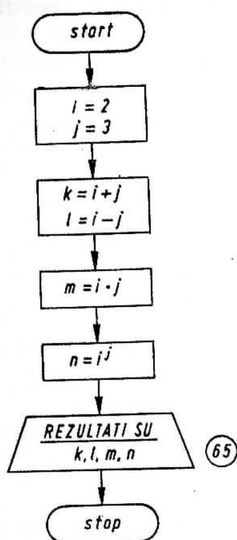
Širina formata mora biti dovoljna ne samo za tiskanje rezultata, već i za bjeline između pojedinih brojčanih vrijednosti (radi preglednosti). Preporučuje se u početku upotrebljavati format I 10.

Preuzak format nema dovoljno mjesta za tiskanje svih znakova rezultata. U tom slučaju će računalo otisnuti zvjezdice u čitavoj širini specificiranog polja.

6. LEKCIJA

RAČUNSKE OPERACIJE

```
      I=2
      J=3
C         ZBRAJANJE I ODBIJANJE
      K=I+J
      L=I-J
C         MNOZENJE
      M=I*J
C         POTENCIRANJE
      N=I**J
C         SLOZENI FORMAT
      WRITE(3,65)K,L,M,N
   65 FORMAT(13H REZULTATI SU,4I5)
      STOP
      END
```



Zadatak je u ovoj lekciji da se izvedu neke osnovne računske operacije, i to zbrajanje, odbijanje, množenje i potenciranje (dijeljenje je ostavljeno za kasnije). Rezultati treba da budu otisnuti zajedno s pripadnim tekstom, tj. u složenom formatu i tekst i brojčani podaci. Postupak je prikazan na shemi toka.

Izlazna lista izgleda ovako:

```
 REZULTATI SU    5   -1    6    8
```


Aritmetičke naredbe mogu u izrazu s desne strane znaka jednakosti sadržavati varijable i konstante, a između njih nalaze se aritmetički operatori.

+ je aritmetički operator za naredbu zbrajanja.
— je aritmetički operator za naredbu odbijanja.
* je aritmetički operator za naredbu množenja.
** je aritmetički operator za naredbu potenciranja.

Aritmetički operatori smiju se upotrebljavati samo svaki za sebe. Ne smiju biti dva ili više operatora zajedno. Isto tako se operator ne smije izostaviti (naročito treba paziti kod množenja).

Složeni format može se koristiti za bilo kakvu kombinaciju teksta i brojčanih podataka.

Maksimalna vrijednost cjelobrojnog rezultata je 32767. Ako bi rezultat računske operacije morao biti veći od toga, računalo će dobiti krivu vrijednost za rezultat.

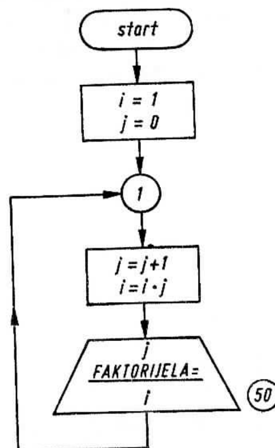
Minimalna vrijednost cjelobrojnog rezultata je — 32767. Ako bi rezultat računske operacije morao biti manji od toga, računalo će dobiti krivu vrijednost za rezultat.

7. LEKCIJA

BEZUVJETNI SKOK — PONAVLJANJE RUTINE

```
C           PRORACUN FAKTORIJELA
      i=1
      J=0
C           POCETAK RUTINE
    1 J=J+1
      i=i*J
      WRITE(3,50)J,i
   50 FORMAT(i3,14H FAKTORiJELA =,i5)
C           SKOK NA POCETAK RUTiNE
      GO TO 1
C           NEMA NAREDBE ZA ZAUSTAVLJANJE
      END
```

Zadatak je ovdje da se izračuna 1! (jedan faktorijel), 2!, 3! itd. za prirodne brojeve redom, i da se to na prikladan način otisne. Postupak za rješenje ovog zadatka prikazan je na shemi toka. Ovdje se već radi o određenom matematičkom postupku, a takve postupke nazivamo algoritmima. Vidi se da se jedna određena rutina ponavlja, i to tako da se prijašnji rezultat množi s brojem koji je svaki puta veći za jedan. Ujedno se to i otisne.



Početak izlazne liste s rezultatima rada računala izgleda ovako:

```
  1 FAKTORIJELA =    1
  2 FAKTORIJELA =    2
  3 FAKTORIJELA =    6
  4 FAKTORIJELA =   24
  5 FAKTORIJELA =  120
  6 FAKTORIJELA =  720
  7 FAKTORIJELA = 5040
```

Algoritam je postupak za rješavanje nekog matematičkog zadatka, a može biti prikazan matematičkim formulama i riječima, ili pomoću sheme toka.

Prirodni redoslijed pri izvršenju programa jest onaj pri kome se naredbe izvršavaju onim redom kako su napisane. Svaka naredba implicitno sadržava i nalog da se nakon njenog izvršenja prijeđe na slijedeću naredbu, osim kod naredaba gdje je ekspilicitno naređeno drugačije.

Saobraćajne naredbe eksplicitno određuju koju naredbu treba izvršiti kao slijedeću, pa prema tome iza njih ne vrijedi više prirodni redoslijed. Saobraćajne naredbe su izvršne naredbe.

GO TO 1 je saobraćajna naredba za bezuvjetan skok. Ona kaže da pri izvršenju programa treba prijeći na izvršenje naredbe s brojem 1, ili općenito da treba prijeći na onaj broj naredbe koji je napisan iza GO TO.

Broj naredbe na koji GO TO upućuje mora postojati u programu, tj. u programu mora postojati naredba s istim brojem naredbe kao što je broj iza GO TO.

Prekrivanje brojčane vrijednosti nastaje kad neka varijabla tokom programa mijenja svoju vrijednost. Nova vrijednost prekrije staru i ostaje zapamćena u memoriji, a stara vrijednost se gubi.

Nova vrijednost može se izračunati koristeći staru vrijednost iste varijable. Na primjer, $j = j + 1$ nije jednadžba (što bi bilo besmisleno), već naredba koja kaže: uzmi (staru) vrijednost od j. povećaj je za 1, i tu dobivenu (novu) vrijednost pridaj varijabli j. Drugim riječima: »neka j bude jednako $j + 1$«, što u stvari znači: »povećaj j za 1«.

Rutina je dio programa koji sačinjava zaokruženu cjelinu.

Bez zaustavljanja može biti program ako ne želimo unaprijed utvrditi koliko puta ćemo ponavljati istu rutinu. U tom slučaju moramo računalo zaustaviti ručno, kad nas daljnje tiskanje više ne interesira (u našem primjeru kad rezultat prekorači vrijednost 32767). Ima, dakako, i boljih rješenja, koja ćemo naučiti kasnije.

Strelica u shemi toka crta se samo za smjer odozdo prema gore (eksplicitno određen smjer), dok se za smjer odozgo prema dolje može izostaviti (smjer prema dolje se implicitno podrazumijeva).

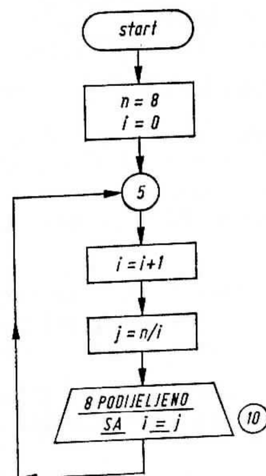
Krug u shemi toka označuje mjesto u programu na koje se dolazi saobraćajnom naredbom. Najbolje je u krug napisati isti broj kao što je odgovarajući broj naredbe u programu.

8. LEKCIJA

DIJELJENJE, TISKANJE TEKSTA

```
      N=8
      i=0
    5 i=i+1
C          DIJELJENJE (CiJELiH BROJEVA)
      J=N/i
C          STAMPANJE (APOSTROFi ZA TEKST)
      WRiTE(3,10)i,J
   10 FORMAT(' 8 PODiJELJENO SA',i3,' =',i2)
      GO TO 5
      END
```

Zadatak je ovdje da se neki broj dijeli redom s prirodnim brojevima, i da se rezultati otisnu. Shema toka pokazuje da je opet upotrijebljeno ponavljanje rutine, slično kao u prijašnjoj lekciji. U ovoj lekciji dolazi do izražaja specifičnost dijeljenja cijelih brojeva, čemu treba posvetiti posebnu pažnju. Pri tiskanju teksta upotrijebljena je nova mogućnost specifikacije u naredbi FORMAT pomoću apostrofa.



Početak izlazne liste s rezultatima izgleda ovako:

```
 8 PODIJELJENO SA  1 = 8
 8 PODIJELJENO SA  2 = 4
 8 PODIJELJENO SA  3 = 2
 8 PODIJELJENO SA  4 = 2
 8 PODIJELJENO SA  5 = 1
 8 PODIJELJENO SA  6 = 1
 8 PODIJELJENO SA  7 = 1
 8 PODIJELJENO SA  8 = 1
 8 PODIJELJENO SA  9 = 0
```

/ je aritmetički operator za naredbu dijeljenja.

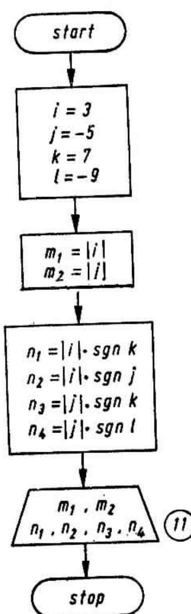
'......' - format (apostrofi u zagradi iza FORMAT) znači da tekst, koji se nalazi unutar apostrofa, treba doslovno otisnuti. Svrha mu je ista kao kod H-formata.

Dijeljenje cijelih brojeva daje kao rezultat opet cio broj. Sve ono što bi bilo iza decimalne točke odbacuje se. Ova specifičnost dijeljenja cijelih brojeva koristi se pri rješavanju određenih zadataka.

9. LEKCIJA

KOMPILATORSKE FUNKCIJE

```
      i=3
      J=-5
      K=7
      L=-9
C          APSOLUTNA VRIJEDNOST
      M1=iABS(i)
      M2=iABS(J)
C          SiGNUM
      N1=iSiGN(i,K)
      N2=iSiGN(i,J)
      N3=iSiGN(J,K)
      N4=iSiGN(J,L)
C          STAMPANJE
      WRiTE(3,11)M1,M2,N1,N2,N3,N4
 11   FORMAT(6i5)
      STOP
      END
```



Zadatak u ovome programu je u tome da se:

1. nađe apsolutna vrijednost varijable;
2. da se apsolutnoj vrijednosti jedne varijable prida predznak druge varijable. To sve treba učiniti u više kombinacija i tiskati rezultate. Shema toka prikazuje odgovarajući postupak.

Izlazna lista s rezultatima izgleda ovako:

```
    3    5    3   -3    5   -5
```

Funkcija se sastoji od imena funkcije i od jednog ili više argumenata u zagradi iza imena funkcije. Argumenti su međusobno odvojeni zarezima. Uz ime funkcije vezana je određena računska operacija s argumentima, tako da se

kao rezultat dobiva jedna (samo jedna) brojčana vrijednost, koja se pridaje toj funkciji i s kojom se može dalje računati.

Kompilatorska funkcija je takva funkcija, kod koje je pripadna računska operacija definirana u samom FORTRAN-kompilatoru, pa takvu funkciju ne treba posebno definirati. Kompilatorsku funkciju može se bez daljnjega pozvati na izvršenje.

Ime funkcije gradi se na isti način kao i imena varijabla. Kod kompilatorskih funkcija imena su već utvrđena FORTRAN-om.

Cjelobrojna funkcija je ona čiji rezultat je cjelobrojna brojčana vrijednost.

IABS (.......) je kompilatorska funkcija za izračunavanje apsolutne vrijednosti argumenta (cjelobrojno).

ISIGN (......., ...) je kompilatorska funkcija čiji je rezultat apsolutna vrijednost prvog argumenta, s predznakom drugog argumenta (cjelobrojno).

Poziv funkcije na izvršenje vrši se tako da se u aritmetičkoj naredbi u izrazu s desne strane znaka jednakosti napiše ime funkcije s pripadnim argumentima.

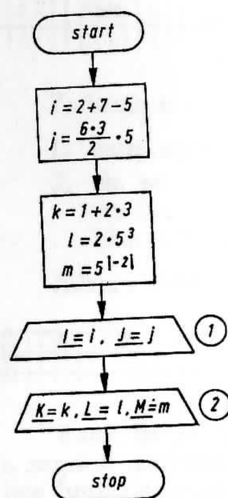
Ime funkcije s argumentima može u aritmetičkoj naredbi stajati samo s desne strane znaka jednakosti, ali može biti i dio većeg aritmetičkog izraza.

Argumenat u pozivu funkcije može biti varijabla, konstanta ili čitav aritmetički izraz.

10. LEKCIJA

HIJERARHIJA ARITMETIČKIH OPERACIJA

```
C          OPERACIJE ISTOG RANGA
      I=2+7-5
      J=6*3/2*5
C          OPERACIJE RAZLICITOG RANGA
      K=1+2*3
      L=2*5**3
      M=5**IABS(-2)
C          STAMPANJE REZULTATA
      WRITE(3,1)I,J
    1 FORMAT(' I =',I5,', J =',I5)
      WRITE(3,2)K,L,M
    2 FORMAT(' K =',I5,', L =',I5,', M =',I5)
      STOP
      END
```



Zadatak je ovdje da se unutar iste aritmetičke naredbe izvrši više računskih operacija, i da se otisnu rezultati. Shema toka prikazuje odgovarajući postupak. Pri tome ćemo naučiti kojim redom se izvršavaju računske operacije unutar iste aritmetičke naredbe.

Izlaznu listu s rezultatima dobit ćemo u slijedećem obliku:

```
I =    4, J =   45
K =    7, L =  250, M =   25
```


Rang pojedinih aritmetičkih operacija ide ovim redom: 1) izračunavanje vrijednosti funkcije; 2) potenciranje; 3) množenje, odnosno dijeljenje i 4) zbrajanje, odnosno odbijanje.

Operacije istog ranga izvršavaju se redom slijeva nadesno. To je implicitni način određivanja redoslijeda. Treba obratiti pažnju pri množenju i dijeljenju (ako množenje dolazi iza dijeljenja, onda treba množiti čitav rezultat, a ne samo nazivnik!).

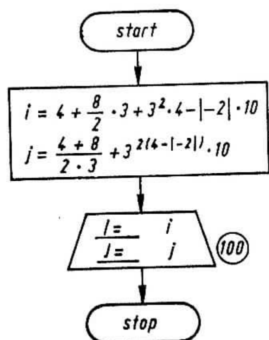
Operacije različitog ranga izvršavaju se takvim redom da se najprije izvrše operacije najvišega ranga, a onda redom operacije sve nižeg ranga. I ovo je implicitni način određivanja redoslijeda.

11. LEKCIJA

ZAGRADE U ARITMETIČKIM IZRAZIMA

```
C         IZRAZ BEZ ZAGRADA
      I=4+8/2*3+3**2*4-IABS(-2)*10
C         IZRAZ SA ZAGRADAMA
      J=(4+8)/(2*3)+(3**(2*(4-IABS(-2))))*10
C         STAMPANJE REZULTATA
      WRITE(3,100)I,J
100   FORMAT(' I = ',I5,', J = ',I5)
      STOP
      END
```

Zadatak je ovdje da se izračunaju dva aritmetička izraza, koji se razlikuju samo u tome što su drugome dodane zagrade. Time se mijenja redoslijed izvršenja pojedinih aritmetičkih operacija, pa se dobivaju različiti rezultati (koje treba otisnuti). Postupak je prikazan na shemi toka.



Izlazna lista s rezultatima dobiva se u ovom obliku:

```
I =    32,J =   812
```


Zagrade u aritmetičkom izrazu eksplicitno određuju redoslijed izvršenja aritmetičkih operacija. Prvo po redu izvršenja dolazi ono što je u zagradi, a tek onda ostalo.

Više zagrada jedna iza druge rješavaju se tako da se riješi svaka zagrada za sebe, a s onim što se dobije računa se redom slijeva nadesno.

Više zagrada jedna u drugoj rješavaju se tako da se najprije riješi ona zagrada koja se nalazi najviše unutra, zatim od zagrada koje su preostale opet ona koja je najviše unutra i tako dalje.

Suvišne zagrade, koje eksplicitno određuju redoslijed, iako je isti redoslijed implicitno određen i bez tih zagrada, ne smetaju i smiju se upotrijebiti.

Količina zagrada, otvorenih i zatvorenih, u istom aritmetičkom izrazu mora biti međusobno jednaka.

12. LEKCIJA

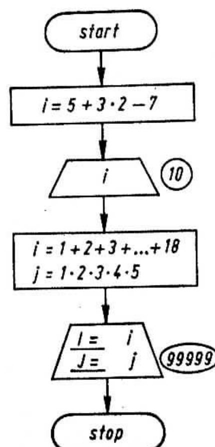
RASPORED U FORMULARU

```
C         SMJESTAJ FORMATA
   10   FORMAT (   i10   )
C         BJELINE i PRAZNi REDAK
       i =  ( 5 + 3 * 2 )   - 7
            WRiTE  ( 3 ,  10   )  i

C         PRODUZENJE NAREDBE U NOVi REDAK
       i  =     1    +    2    +    3
     N    +     4    +    5    +    6
     7    +     7    +    8    +    9
     *    +    10    +   11    +   12
     P    +    13    +   14    +   15
     5    +    16    +   17    +   18

     0  J     =    1    *    2    *    3
     1        *    4    *    5

       WRiTE(3,99999)i,J
       STOP
99999  FORMAT(' i = ',i5,',J = ',i5)
                        END
```



Zadatak je ovdje da se izračunaju neki aritmetički izrazi, ali je zapravo svrha da se vidi kako se upotrebljava FORTRAN-formular. Važno je znati čega se moramo pridržavati pri pisanju programa, te koju slobodu pri tome imamo. Matematički postupak prikazan je u shemi toka.

Izlazna lista s rezultatima rada računala ovog je oblika:

```
I =   171,J =   120
```


Bjeline se pri pisanju programa smiju umetati po želji, tj. kućice u formularu smiju se ostavljati prazne. To se koristi ako se želi povećati preglednost napisanog programa.

Prazni redak pri pisanju programa smije se također ostavljati po želji (to u stvari znači jednu praznu karticu).

Produženje naredbe u nov redak vrši se tako da se u stupac 6 upiše bilo koji znak, osim nule (ne smije se niti ostaviti bjelina). Produženja smije biti najviše pet. Produženje naredbe rabi se ako naredba ne može stati u jedan redak, ili zbog preglednosti.

Prvi redak produžene naredbe mora u koloni 6 imati nulu ili bjelinu.

Naredbe FORMAT smiju se smjestiti bilo gdje unutar programa (što, dakako, znači da ne smiju biti iza naredbe END). To je dopušteno zbog toga što je FORMAT organizaciona naredba, vezana uz pripadnu izvršnu naredbu WRITE. Uostalom više naredaba WRITE može se pozivati na isti FORMAT.

Stupac 1 u formularu služi za upisivanje slova C kao organizacione naredbe za komentar.

Stupac 1 do 5 u formularu služe za upisivanje broja naredbe. Broj naredbe smije se unutar tih pet stupaca smjestiti bilo kako.

Stupac 6 u formularu služi za upisivanje oznake kod produženja naredbe.

Stupci 7 do 72 u formularu služe za upisivanje naredbe.

Stupci 73 do 80 nisu obuhvaćeni FORTRAN-om i kompilator ih neće uzeti u obradu. Ti se stupci mogu upotrijebiti za identifikaciju (ime programa i broj kartice).

13. LEKCIJA

UVJETNI SKOK — RAČVANJE

```
      M=15
      N=-2
C     POCETAK RUTINE
    1 N=N+1
C     BIRANJE POSTUPKA
      IF(N)2,3,4
    2 WRITE(3,12)M,N
      GO TO 1
    3 WRITE(3,13)M,N
      GO TO 1
C     ISPITIVANJE DJELJIVOSTI
    4 IF(M-M/N*N)5,5,6
    5 WRITE(3,15)M,N
      GO TO 1
    6 WRITE(3,16)M,N
      GO TO 1
   12 FORMAT(I5,'/',I2,' DJELITELJ NEGATIVAN')
   13 FORMAT(I5,'/',I2,' DJELITELJ JE NULA')
   15 FORMAT(I5,'/',I2,' DJELJIVO')
   16 FORMAT(I5,'/',I2,' NIJE DJELJIVO')
      END
```

Zadatak je ovdje da se ispita da li je neki zadani broj djeljiv s pojedinim članovima iz niza brojeva i da se ovisno o rezultatu otisne odgovarajuća obavijest. Niz brojeva, koji će predstavljati djelitelje, treba generirati u samom programu, povećavajući djelitelj uvijek za jedan. Traži se posebna obavijest ako je djelitelj negativan, i posebna ako je jednak nuli. Ako je pak djelitelj pozitivan, treba ispitati postoji li djeljivost i dati obavijest da li je djeljivo ili nije. Očito je da će postupak trebati ponavljati za svaki novi djelitelj, ali će u ponavljanju postojati različite staze ovisno o rezultatu ispitivanja: 1) da li je djelitelj negativan, nula ili pozitivan i 2) da li kod pozitivnog djelitelja djeljivost postoji ili ne. Hoćemo li tokom izvršenja programa skočiti na ovu ili onu stazu, uvjetovano je rezultatom ispitivanja, pa nam je ovdje potreban uvjetan skok. Ipak u ovom slučaju moramo koristiti i bezuvjetan skok, jer imamo ponavljanje rutine. Prikaz toka za ovaj postupak je već malo složeniji, ali jasno pokazuje kako treba raditi.

Prilikom izvršenja ovog programa dobiva se ovakva lista s rezultatima:

```
   15/-1 DJELITELJ NEGATIVAN
   15/ 0 DJELITELJ JE NULA
   15/ 1 DJELJIVO
   15/ 2 NIJE DJELJIVO
   15/ 3 DJELJIVO
   15/ 4 NIJE DJELJIVO
   15/ 5 DJELJIVO
   15/ 6 NIJE DJELJIVO
   15/ 7 NIJE DJELJIVO
   15/ 8 NIJE DJELJIVO
   15/ 9 NIJE DJELJIVO
   15/10 NIJE DJELJIVO
   15/11 NIJE DJELJIVO
   15/12 NIJE DJELJIVO
   15/13 NIJE DJELJIVO
   15/14 NIJE DJELJIVO
   15/15 DJELJIVO
   15/16 NIJE DJELJIVO
   15/17 NIJE DJELJIVO
   15/18 NIJE DJELJIVO
   15/19 NIJE DJELJIVO
   15/20 NIJE DJELJIVO
```

IF (.......) ...., ..., ... je naredba za uvjetan skok, koji je uvjetovan brojčanom vrijednošću izraza u zagradi za IF. Ovo je dakako **izvršna naredba.**

Brojčana vrijednost izraza u zagradi iza IF uvjetuje skok na jednu od naredbi, čiji se brojevi naredbi nalaze iza zagrade, i to ovako:

ako je **manja od** nule . . . . . skače se na **prvi** broj naredbe,
ako je **jednaka** nuli . . . . . skače se na **drugi** broj naredbe,
ako je **veća od** nule . . . . . skače se na **treći** broj naredbe.

Dvodimenzionalnost sheme toka dolazi ovdje do izražaja, jer su pojedine staze nacrtane jedna pokraj druge. Time se znatno povećava preglednost sheme toka. Zbog toga pri crtanju sheme toka treba uvijek (kad za to postoji prilika) koristiti obje dimenzije papira (visinu i širinu).

Jednodimenzionalnost jest jedno od svojstava programa, jer se u njemu naredbe moraju pisati jedna iza druge. Zbog toga se i čitave staze iz sheme toka moraju u program prenositi jedna iza druge. Njihov raspored u programu je proizvoljan, ali na kraju svake staze mora biti utvrđeno s kojom naredbom se program nastavlja.

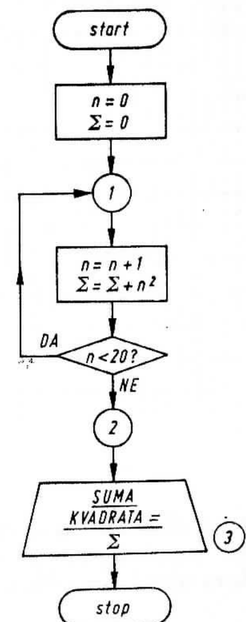
Računanje s cijelim brojevima dolazi u ovom slučaju do izražaja pri izračunavanju ostatka kod dijeljenja (koji je korišten kao kriterij za djeljivost). Vrijednost izraza u zagradi kod naredbe broj 4 računa se npr. ovako:
15 — 15/2*2 = 15 — 7*2 = 15 — 14 = 1.

Romb u shemi toka se upotrebljavaju za uvjetnu naredbu IF.

LEKCIJA **14.**

UVJETNI SKOK — PREKID PONAVLJANJA

```
C          SUMA KVADRATA
      N=0
      KVASU=0
C          POCETAK RUTiNE
    1 N=N+1
      KVASU=KVASU+N**2
C          PONAVLJANJE DO N=20
      iF(N-20)1,2,2
C          STAMPANJE REZULTATA
    2 WRiTE(3,3)KVASU
      STOP
    3 FORMAT(' SUMA KVADRATA =',i5)
      END
```



Zadatak je ovdje da se izračuna suma kvadrata prvih 20 prirodnih brojeva. Algoritam za sumu je jednostavan: u ponavljanoj rutini se već izračunatoj sumi svaki put dodaje novi sumand. U našem zadatku taj sumand je kvadrat broja, koji se svaki put povećava za jedan. Kad sumand postigne vrijednost 20, ponavljanje treba prekinuti i otisnuti rezultat. Shema toka prikazuje opisani postupak.

Izlazna lista s rezultatima izgleda ovako:

```
SUMA KVADRATA = 2870
```

Izraz u zagradi iza IF može biti općenito bilo koji aritmetički izraz. Taj izraz može degenerirati na samu jednu varijablu.

IF kao naredba za povrat na početak rutine upotrebljena je ovdje dok je ispunjen prvi uvjet, tj. $n - 20$ je manje od nule (to u stvari znači da je n manje od 20).

IF kao naredba za prekid ponavljanja i za prijelaz na slijedeći dio programa, upotrijebljena je ovdje za slučaj kad je ispunjen drugi uvjet, tj. $n - 20$ je jednako nuli (to u stvari znači da je n jednako 20).

Slijepi broj naredbe (treći u naredbi IF) upotrijebljen je iako treći uvjet, tj. $n - 20$ je veće od nule (to u stvari znači da je n veće od 20), nikad neće biti ispunjen.

Tri broja naredbe moraju biti u naredbi IF. Po potrebi stavlja se slijepi broj naredbe, i to jedan od onih koji su već upotrijebljeni u toj naredbi IF, obično onaj koji je manje nelogičan. U našemu primjeru, kad bi n bio veći od 20, ne bi trebalo ponavljati rutinu, isto kao kad je $n = 20$.

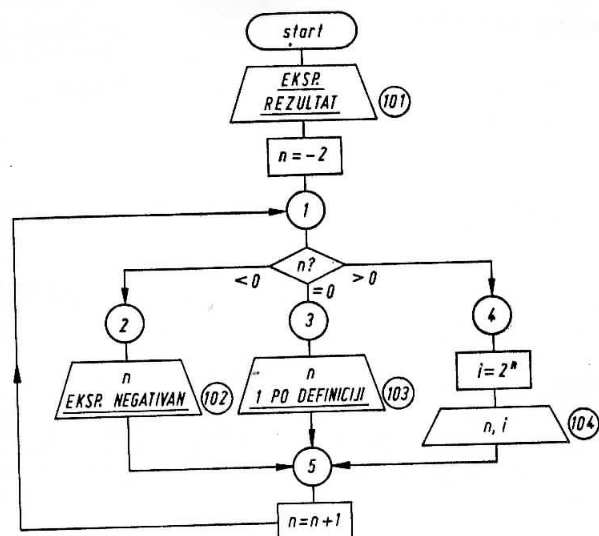
LEKCIJA **15.**

SASTAJANJE STAZA

Zadatak je ovdje da se izračunaju potencije broja 2, pri čemu je eksponent u početku negativan, ali je u svakom ponovljenom računu povećan za jedan. Dok eksponent nije pozitivan treba kao rezultat otisnuti odgovarajuću obavijest, a kod pozitivnih eskponenata treba otisnuti rezultate. Sve to treba dati u formi tabele.

```
1   5 6 7  10        20        30        40        50        60
C            POTENCIRANJE BROJA 2
      WRITE(3,101)
      N=-2
1     IF(N)2,3,4
2     WRITE(3,102)N
      GO TO 5
3     WRITE(3,103)N
      GO TO 5
4     I=2**N
      WRITE(3,104)N,I
5     N=N+1
      GO TO 1
101   FORMAT(' EKSP.REZULTAT')
102   FORMAT(I4,'   EKSPONENT NEGATIVAN')
103   FORMAT(I4,'      1 PO DEFINICIJI')
104   FORMAT(I4,I6)
      END
```

Postupak je prikazan u shemi toka, koja je sastavljena tako da se tri staze sastaju i nastavak je zajednički. Na tom mjestu moramo imati broj naredbe (5 u krugu) na koji će upućivati saobraćajna naredba pri kraju svake staze.



Izlazna lista s rezultatima izgleda ovako (početak):

```
EKSP.  REZULTAT
 -2     EKSPONENT NEGATIVAN
 -1     EKSPONENT NEGATIVAN
  0         1 PO DEFINICIJI
  1         2
  2         4
  3         8
  4        16
  5        32
  6        64
  7       128
  8       256
  9       512
 10      1024
 11      2048
 12      4096
 13      8192
 14     16384
```

Staze koje se sastaju moraju na kraju imati naredbu GO TO, koja upućuje na nastavak programa koji je zajednički iza paralelnih staza. Naredba GO TO može se izostaviti kod staze koja je posljednja napisana u programu, jer je u posljednjoj naredbi implicirana naredba za nastavak (a to je upravo zajednički dio programa).

Krug s brojem naredbe mora u shemi toka doći: 1) na mjesto kamo se vrši povratni skok, 2) na početak staza iza račvanja i 3) na spojište paralelnih staza.

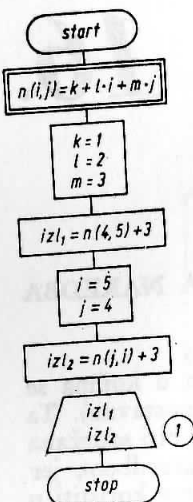
LEKCIJA **16.**

FUNKCIJSKA NAREDBA

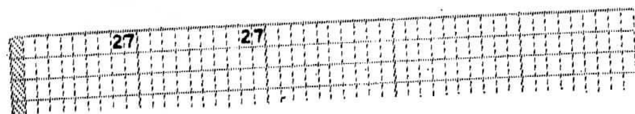
Zadatak je ovdje da se izračuna vrijednost nekoliko izraza u kojima se pojavljuje jedna te ista funkcija (proizvoljna, ali dovoljno jednostavna). Ta se funkcija najprije mora definirati, jer je proizvoljna, pa ne može biti sadržana u kompilatoru. U ovom slučaju je funkcija definirana jednom naredbom, jer je dovoljno jednostavna pa je to moguće. Sad ovu funkciju možemo koristiti u programu na isti način kao što smo to radili s kompilatorskim funkcijama.

```
1   5 6 7  10        20        30        40        50        60
C            PRIMJER ZA FUNKCIJSKU NAREDBU
C            DEFINICIJA FUNKCIJSKE NAREDBE
        N(I,J)=K+L*I+M*J
C            BROJCANA VRIJEDNOST PARAMETRA
        K=1
        L=2
        M=3
C            POZIV (ARGUMENTI SU KONSTANTE)
        IZL1=N(4,5)+3
C            BROJCANA VRIJEDNOST ARGUMENATA
        I=5
        J=4
C            POZIV (ARGUMENTI SU VARIJABLE)
        IZL2=N(J,I)+3
             STAMPANJE
        WRITE(3,1)IZL1,IZL2
    1   FORMAT(2I10)
        STOP
        END
```

Shema toka je ovdje posve jednostavna, jer je i postupak unutar programa jednostavan. Svrha ove lekcije je ionako samo ta da nas uvede u funkcije naredbe i da to ilustrira.



Izlazna lista s rezultatima izgleda ovako:



Potprogram je posebno organizirana rutina, tako da je u programu možemo proizvoljno ponavljati kad god želimo. Za razliku od toga, ponavljanje rutine pomoću GO TO ili IF vrši se pravilno određenim redom.

Dvostruki pravokutnik u shemi toka upotrebljava se za označavanje potprograma.

Funkcijska naredba je takav potprogram gdje se dobiva samo jedan rezultat (to vrijedi općenito za funkcije, od kojih smo već prije upoznali kompilatorske funkcije), a potrebne računske operacije za dobivanje tog rezultata definirane su u samoj jednoj naredbi.

Definicija funkcijske naredbe je organizaciona naredba, a mora stajati na početku programa prije prve izvršne naredbe. S lijeve strane znaka jednakosti stoji ime funkcije s jednim ili više argumenata u zagradi, a s desne strane stoji aritmetički izraz za izračunavanje vrijednosti funkcije.

Ime funkcije gradi se na isti način kao imena varijabla.

Formalni argumenti nalaze se u definiciji funkcijske naredbe uz ime funkcije. Njihova imena nemaju nikakve veze s imenima izvan definicije, već samo služe za to da unutar funkcijske naredbe definiraju postupak za izračunavanje vrijednosti funkcije.

Poziv funkcije vrši se unutar programa tako, da se u odgovarajućoj izvršnoj naredbi navede ime funkcije, a uz nju stvarni argumenti. Ako to činimo unutar neke aritmetičke naredbe, očito je da će ime funkcije sa stvarnim argumentima stajati s desne strane znaka jednakosti.

Stvarni argumenti nalaze se u pozivu funkcije, a mogu biti konstante, varijable ili općenito aritmetički izrazi. S brojčanom vrijednošću tih argumenata računa se vrijednost funkcije. Imena stvarnih argumenata nemaju nikakve veze s imenima formalnih argumenata, važan je samo njihov redoslijed.

Poklapanje po broju i redoslijedu mora postojati između formalnih argumenata u definiciji i stvarnih argumenata u pozivu.

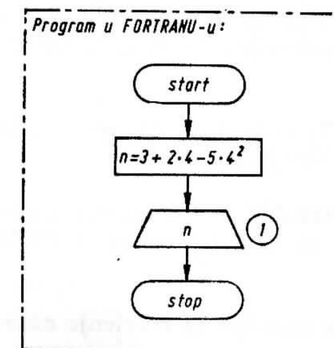
Parametri su varijable s desne strane znaka jednakosti u definiciji funkcijske naredbe, koji se ne nalaze u zagradi uz ime funkcije s lijeve strane znaka jednakosti. Parametri su vezani svojim imenom na iste varijable u programu izvan definicije funkcijske naredbe. Brojčana vrijednost parametara mora u programu biti utvrđena prije poziva funkcije.

LEKCIJA **17.**

MONITOR

```
1   5 6 7  10        20        30        40
// JOB T           STEFANINI
// FOR
*IOCS(1132PRINTER)
*LIST SOURCE PROGRAM
C           VRIJEDNOST POLINOMA (FORTRAN)
      N=3+2*4-5*4**2
      WRITE(3,1)N
    1 FORMAT(I10)
      CALL EXIT
      END
// XEQ
```

Zadatak je ovdje veoma jednostavan u matematičkom pogledu, i to zbog toga da što jače iskoči ono što je važno u ovoj lekciji, a to su monitorske naredbe. Ako računalo nema MONITOR, onda pojedine radnje na računalu mora vršiti operater. Ako pak računalo ima MONITOR, onda se taj posao pomoću odgovarajućih naredaba prenese na samo računalo. To je i učinjeno u ovom programu. Shema toka je proširena monitorskim naredbama, tako da u potpunosti odgovara ispisanom programu. Inače je običaj da se shema toka crta samo za one naredbe koje će se ispisivati u FORTRAN-u (kao što smo do sada činili).



Potpuna izlazna lista s otisnutim monitorskim naredbama i sa cijelim programom u FORTRAN-u (jer smo to zahtijevali), te konačno s rezultatima, izgleda ovako:

```
// JOB T               STEFANINI
// FOR
*IOCS(1132PRINTER)
*LIST SOURCE PROGRAM
C            VRIJEDNOST POLINOMA (FORTRAN)
      N=3+2*4-5*4**2
      WRITE(3,1)N
    1 FORMAT(I10)
      CALL EXIT
      END

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON        0  VARIABLES         4  PROGRAM      54

END OF COMPILATION
// XEQ

       -69
```

MONITOR je poseban sistem, koji može koristiti računalo ako ima i memoriju na disku. Upotrebom monitora računalo samo vrši mnoge poslove umjesto operatera, pa se time uvelike ubrzava rad.

Monitorska naredba je naredba kojom se daju određene upute MONITOR-u. Za to pisanje se obično koristi isti formular kao i za FORTRAN, no treba paziti da se sve piše točno po stupcima, kako je propisano.

// JOB T je monitorska naredba za izvršenje posla privremena karaktera. Oznaka T označava privremeni karakter, tako da nakon završetka tog posla ništa neće ostati trajno spremljeno na disku. Prostor iza 20. stupca može se proizvoljno koristiti za komentar (ime korisnika i slično).

// FOR je monitorska naredba za pozivanje FORTRAN-kompilatora sa diska u glavnu memoriju.

Fortranska upravljačka naredba je naredba kojom se daju upute što sve treba učiniti prilikom kompiliranja. Te se naredbe moraju nalaziti iza monitorske naredbe // FOR.

***IOCS (1132 PRINTER)** je fortranska upravljačka naredba za pozivanje rutina potrebnih linijskoj tiskaljki.

***LIST SOURCE PROGRAM** je fortranska upravljačka naredba za tiskanje svih kartica na kojima se nalazi program u FORTRAN-u, i to prilikom kompiliranja.

// XEQ je monitorska naredba za izvršenje netom učitanog i kompiliranog programa.

CALL EXIT je izvršna naredba unutar programa pisanog u FORTRAN-u, koja naređuje završavanje toga programa i nastavak rada računala prema daljnjim monitorskim naredbama. CALL EXIT se smije upotrijebiti samo onda ako računalo radi s MONITOROM. CALL EXIT se redovito upotrebljava umjesto naredbe STOP, čime uštedjuje vrijeme, jer se računalo ne zaustavlja u radu.

Kompilatorska obavijest je obavijest koju računalo otisne prilikom kompiliranja. Računalo uvijek otisne slijedeće tri kompilatorske obavijesti.

FEATURES SUPPORTED je kompilatorska obavijest o tome koje je zahtjeve računalo uvažilo prilikom kompiliranja.

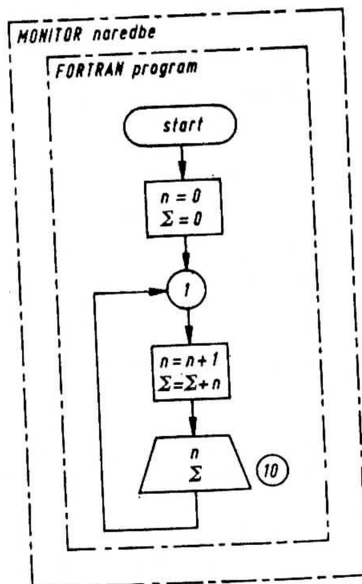
CORE REQUIREMENTS FOR je kompilatorska obavijest o broju ćelija memorije, koliko je za pojedinu svrhu zauzeto prilikom kompiliranja.

END OF COMPILATION je kompilatorska obavijest o tome da je kompiliranje završeno.

18. LEKCIJA

POGREŠKE U FORTRANU

```
// JOB T                 STEFANINI
// FOR
*IOCS(1132PRINTER)
*LIST SOURCE PROGRAM
C           SUMA PRIRODNIH BROJEVA
      N=0
      ISUM=0
1     N=N+
      ISUM=ISUM+M
      WRITE(3,10),N,ISUM
10    FORMAT(2I10)
5     GO TO 1
      STOP
      END
// XEQ
```



Zadatak u matematičkom pogledu je ovdje posve jednostavan i njime se nećemo baviti, jer je svrha ove lekcije sasvim posebna. Želimo pokazati što se događa ako se pojedine naredbe ispišu pogrešno, ili ako se u programu nalaze neke sintaktičke pogreške, koje ometaju ispravno kompiliranje programa. Zbog toga je ovdje program namjerno ispisan s pogreškama, a ne kako bi bilo ispravno i kako bi odgovaralo slijedećoj shemi toka (koja je u redu).

FORTRAN-kompilator je tako organiziran da prilikom učitavanja programa ujedno analizira eventualne pogreške i o tome daje odgovarajuću obavijest. Pri tome kompilator razlikuje 73 različite vrste sintaktičkih pogrešaka. Dakako da se program koji sadrži pogreške ne može kompilirati, pa prema tome niti izvesti, te će i MONITOR dati

odgovarajuće obavijesti u tom smislu. Izlazna lista za gornji program izgledat će ovako (ovdje su ujedno rukom uneseni potrebni ispravci, koji slijede iz analize dobivenih obavijesti):

```
// JOB T                 STEFANINI
// FOR
*IOCS(1132PRINTER)
*LIST SOURCE PROGRAM
C           SUMA PRIRODNIH BROJEVA
      N=0
      ISUM=0
1     N=N+1
      ISUM=ISUM+N
      WRITE(3,10) N,ISUM
10    FORMAT(2I10)
5     GO TO 1
      STOP
      END

UNREFERENCED STATEMENTS
 5

UNDEFINED VARIABLES
 M

INVALID STATEMENTS
 C 36 ERROR AT STATEMENT NUMBER 1
 C 47 ERROR AT STATEMENT NUMBER 1         +002
 C 06 ERROR AT STATEMENT NUMBER 5         +001

OUTPUT HAS BEEN SUPPRESSED

END OF COMPILATION

M 01 PHASE NONX
// XEQ
M 03 NON XEQ
```

Pogreške u programu pronaći će FORTRAN-kompilator prilikom učitavanja programa, te će računalo otisnuti odgovarajuće obavijesti.

UNREFERENCED STATEMENTS je popis brojeva naredbi na koje se u programu nigdje ne pozivamo, pa su ti brojevi naredbi suvišni. Suvišne brojeve naredbi možemo ostaviti u programu, jer to neće ometati kompiliranje i izvršenje programa. Ali ta obavijest može biti posljedica pogreške, pa to treba ispraviti.

UNDEFINED VARIABLES je popis nedefiniranih varijabla, koje se pojavljaju u aritmetičkim izrazima (s jedne strane znaka jednakosti, u naredbi IF, kao stvaran argument u pozivu funkcije), a da im prije toga nije utvrđena brojčana vrijednost.

INVALID STATEMENTS je popis neispravnih naredaba. Za svaku neispravnu naredbu računalo će otisnuti šifru pogreške, tekst ERROR AT STATEMENT NUMBER (što znači: pogreška u naredbi broj), te broj naredbe (ukoliko dotična naredba ima svoj broj), odnosno položaj neispravne naredbe s ozbirom na prethodnu naredbu s brojem.

OUTPUT HAS BEEN SUPPRESSED je obavijest računala da kompiliranje nije do kraja izvršeno.

Popis pogrešaka u FORTRAN-u obuhvaća 73 različite sintaktičke pogreške sa šifrom i značenjem (vidi prilog 2!).

M 01 PHASE NONX je monitorska obavijest da se taj dio JOB-a ne može provesti.

M 03 NON XEQ je monitorska obavijest da program pozvan na izvršenje neće biti izvršen.

Ispravci pogrešaka vrše se na izlaznoj listi, i to pomoću oznaka uobičajenih u tiskarskoj tehnici.

ZAKLJUČAK PRVOG POGLAVLJA

U prvom poglavlju stekli smo već određeno zaokruženo znanje. Naučili smo uglavnom sve osnovne stvari (komentar, raspored u formularu i sl.), uglavnom sve o aritmetičkim izrazima i aritmetičkim naredbama, glavne saobraćajne naredbe, ono što je jednostavno o potprogramima (kompilatorske funkcije i funkcijske naredbe), odnosno o tiskanju teksta i brojčanih vrijednosti, početke upotrebe MONITOR-a i korištenje obavijesti o pogreškama.

Stečeno znanje zasada je ograničeno na cjelobrojne varijable i cjelobrojne konstante. No ipak mnogo toga vrijedi općenito i za slijedeća poglavlja, gdje nećemo raditi samo s cijelim brojevima, već i s realnim brojevima. Ono što za realne brojeve eventualno bude drugačije od dosada naučenoga bit će eksplicitno rečeno, u protivnom implicitno će vrijediti i za realne brojeve ono što smo već naučili.

Naše dosada stečeno znanje ograničeno je još u jednom pogledu. Brojčane vrijednosti koje slijede iz zadatka pridane su pojedinim varijablama unutar samog programa (pomoću aritmetičke naredbe: varijabla = konstanta), ili su u obliku konstanata na neki drugi način upotrijebljene u programu. Na taj način program, dakako, ne može biti općenit, nego vrijedi samo za zadane brojčane podatke. (U drugom poglavlju ćemo se osloboditi ovog ograničenja na taj način što ćemo uvesti učitavanje podataka, pa brojčane vrijednosti neće biti potrebno ugraditi u sam program).

Na kraju ovog poglavlja nalaze se zadaci koji su prilagođeni stečenom znanju i koji se uz pomoć toga znanja mogu riješiti. Zbog ograničenog znanja, stečenog u prvom poglavlju, ti zadaci imaju posebnu fizonomiju. Prvo što je za njih tipično jeste to da se računa samo s cijelim brojevima. Drugo što je tipično jest da izrađeni programi neće biti općeniti, nego će vrijediti samo za zadane brojčane podatke. U praksi, dakako, želimo sastaviti programe koji su posve općeniti, ali ćemo se ovdje (gdje nam je glavna svrha da nešto naučimo) zadovoljiti s takvim zadacima koji nisu tipični za praksu.

Zadaci su podijeljeni u dvije grupe. U grupi A su oni koji ne sadržavaju brojčane podatke. U programu, koji rješava takav zadatak, nalazit će se, dakako, konstante s određenim brojčanim vrijednostima, ali su te vrijednosti dane samom prirodom zadatka, i takav se program ne bi niti mogao načiniti općenitijim. Ovakvi se zadaci mogu pojaviti u praksi i njihovo rješenje, koristeći jedino znanje iz materije prvog poglavlja, može u potpunosti zadovoljiti.

U grupi B nalaze se oni zadaci koji sadržavaju i brojčane podatke. Ovdje ćemo (nemajući još sve potrebno znanje) nužno morati te brojčane podatke ugraditi u program, koji na taj način neće biti općenit. Ova grupa zadataka, prema tome, nije tipična za korištenje elektroničkih računala u praksi, nego nam služi samo za uvježbavanje, kako da koristimo stečeno znanje.

Riješene zadatke u obliku napisanih programa treba dati elektroničkom računalu na izvršenje. To je jedini korisni način da provjerimo jesmo li zadatke ispravno riješili.

ZADACI UZ PRVO POGLAVLJE

Grupa A

1. Tiskaj redom prirodne brojeve (zaustavljanje rukom).
2. Tiskaj redom parne brojeve.
3. Tiskaj redom kvadrate prirodnih brojeva.
4. Tiskaj redom kubuse prirodnih brojeva.
5. Tiskaj redom prirodne brojeve, te kraj njih njihove kvadrate i kubuse. Tiskaj i glavu tablice.
6. Tiskaj redom prirodne brojeve i kraj svakog sumu svih prirodnih brojeva do tog broja.
7. Tiskaj redom prirodne brojeve i kraj svakog umnožak svih prirodnih brojeva do tog broja.
8. Tiskaj redom vrijednosti pozicija u binarnom sistemu (dva na nultu, dva na prvu, dva na drugu itd.).
9. Tiskaj tablicu binomnih koeficijenata.
10. Tiskaj redom sve prim-brojeve.
11. Tiskaj tablicu množenja.
12. Tiskaj tablicu pretvaranja engleskog novca u dinare (1 funta = 20 šilinga = 240 pensa = 3000 starih dinara).
13. Pokušavanjem nađi bar jedan pravokutni trokut s cjelobrojnim stranicama.
14. Pokušavanjem riješi zadatak $x^y = y^x$, $x \neq y$, rješenje u cijelim brojevima.
15. Koliko sedmica ima u prestupnoj godini? Traži se rezultat na tri decimale. Primijeni srednjoškolski postupak dijeljenja.

Grupa B

16. do 21. Isto kao zadaci 1. do 6. ali s tim, da se računalo samo zaustavi kad prirodni broj dosegne vrijednost 20.
22. Isto kao zadatak 7., ali do prirodnog broja 7.
23. Isto kao zadatak 8., ali za 15 pozicija.
24. Isto kao zadatak 9., ali samo 10 redaka.
25. Isto kao zadatak 10., ali samo do 50.
26. Tiskaj redom brojeve djeljive sa 3 (u prvom stupcu) i brojeve djeljive sa 7 (u drugom stupcu).
27. Na »maloj šahovskoj ploči« 3 × 3 polja, na prvom polju nalazi se 1 zrno, na drugom 2 zrna, na trećem 4 i itd. Koliki je ukupni broj zrna?
28. Odredi najveću zajedničku mjeru dvaju brojeva (npr. za 40 i 12).
29. Odredi najmanji zajednički višekratnik dvaju brojeva (npr. za 40 i 12).
30. Rastavi broj na proste faktore (npr. 5880).
31. Odredi cjelobrojna rješenja Diofantove jednadžbe $3x - 17y = 1$ za $y = 1$ do 50.
32. Zadan je broj sa 5 znamenaka (npr. 15346). Otisni njegovu srednju znamenku.

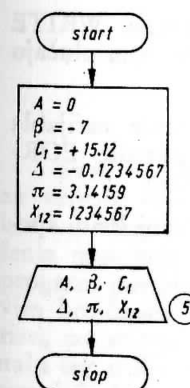
33. Zadan je broj s neparnim brojem znamenaka, ali najviše sa 5 znamenaka (npr. 3 ili 534 ili 15346). Otisni njegovu srednju znamenku.
34. Tiskaj sve brojeve između 200 i 300, koji su djeljivi sa 7.
35. Tiskaj kalendar počev od prvog slijedećeg ponedjeljka.
36. Odredi koliko dana ima između dva različita datuma (na pr. od vlastitog rođendana do danas).

LEKCIJA **19.**

REALNE VELIČINE

```
C                           REALNE VELICINE
C             PRIDAVANJE BROJCANE VRIJEDNOSTI
      A=0.
      BETA=-7.0
      C1=+15.12
      DELTA=-.1234567
      PI=3.14159
      X12=1234567.
C             STAMPANJE BROJCANE VRIJEDNOSTI
      WRITE(3,5)A,BETA,C1,DELTA,PI,X12
    5 FORMAT(4F10.4)
      CALL EXIT
      END
```

Zadatak u ovoj lekciji je da se nekim veličinama pridaju brojčane vrijednosti i da se zatim njihove brojčane vrijednosti otisnu. Svrha je, međutim, u tome da se na taj način uvedu realne veličine, za razliku od cjelobrojnih (koje smo jedine upotrebljavali u prvom poglavlju). Realne veličine razlikuju se od cjelobrojnih u tome što nisu zaokružene na cio broj, već mogu imati decimale. Bitno je kod realnih veličina, međutim, to da njihova brojčana vrijednost nije dana potpuno točno, već samo na određeni broj znamenaka. Zbog toga, ako radimo s realnim veličinama, niti rezultati ne će biti savršeno točni, nego će im točnost biti ograničena. Ipak se u tehnici redovito računa s realnim brojevima. jer točnost dobivenih rezultata zadovoljava. Cjelobrojne veličine obično se upotrebljavaju samo onda kad priroda tih veličina to zahtijeva (npr. za indekse, za brojenje i slično).



Izlazna lista s rezultatom izvršenja ovog programa izgledala bi ovako:

```
    0.0000    -7.0000    15.1200    -0.1234
    3.1415**********
```

Realna veličina definirana je samo s ograničenom točnošću, u našem slučaju normalno na 7 znamenaka. Nema svrhe niti u programu pisati niti u rezultatima otisnuti veći broj znamenaka od toga.

Simboli za realne varijable moraju početi slovom, ali to slovo ne smije biti I, J, K, L, M ili N (što je rezervirano za cjelobrojne varijable). Početnim slovom (A do H, odnosno O do Z) implicitno je određen realan tip varijable. Inače vrijedi isto kao kod cjelobrojnih varijabla (najviše 5 znakova, prvi znak mora biti slovo, ostali znakovi mogu biti i brojke).

Simbole u shemi toka pisat ćemo velikim slovima za realne varijable, a malim slovima za cjelobrojne varijable.

Realna konstanta (brojčana vrijednost unutar programa) mora se uvijek pisati s decimalnom točkom, bez obzira da li pišemo i decimale ili ne. (Pazi: za odvajanje decimalnih mjesta upotrebljava se točka, a ne zarez!).

Nula ispred decimalne točke se kod konstanata u programu može pisati, ali ne mora.

F-format se upotrebljava za tiskanje realnih brojčanih vrijednosti. Prvi broj iza F određuje širinu polja za tiskanje (ukupni broj kućica). Drugi broj (iza točke) određuje koliko decimala će biti otisnuto.

Tiskanje u F-formatu vrši se pozicionirano udesno. Decimalna točka se uvijek otisne. Ukoliko nema cijelih mjesta, ispred decimalne točke otisne se nula. Pozitivni predznak se ne tiska.

Broj decimala specificiran formatom za tiskanje uvijek se ispunjava. Ako je brojčana vrijednost prekratka, otiskuju se nule. Ako je brojčana vrijednost preduga, suvišni broj decimala se odbacuje (pri tome se ne zaokružuje!).

Ponavljanje formata nastaje kad ima više varijabla u naredbi WRITE nego što ima specifikacija za tiskanje u naredbi FORMAT. U tom slučaju format se ponavlja, i to u novom retku.

Neiskorišteni format dolazi do izražaja onda kad ima manje varijabla u naredbi WRITE nego što ima specifikacija za tiskanje u naredbi FORMAT. U tom slučaju suvišni broj specifikacija ostaje neiskorišten.

Preuzak format i ovdje dovodi do tiskanja zvjezdica u čitavoj širini specificinanog polja.

LEKCIJA **20.**

UČITAVANJE ULAZNIH PODATAKA

```
C         UCiTAVANJE PODATAKA
      READ(2,15)A,B,C,D,E,F
   15 FORMAT(3F10.4)
C         STAMPANJE
      WRiTE(3,15)A,B,C,D,E,F
      CALL EXiT
      END
```

Zadatak je ovdje da se učita nekoliko realnih brojčanih vrijednosti, da se te vrijednosti pridaju odgovarajućim varijablama, i da se konačno brojčane vrijednosti tih varijabla tiskaju. Zadatak je u matematičkom pogledu posve jednostavan (u stvari računanja niti nema), pa je i shema toka posve jednostavna.

Svrha je ove lekcije prvenstveno ta, da se uvede učitavanje podataka za vrijeme izvršenja programa. Na taj način brojčane vrijednosti ne moraju biti ugrađene u program (kao konstante u programu), nego se prilikom izvršenja programa mogu u računanje uvesti bilo koje brojčane vrijednosti. Time program postaje općenit, pa se može izvršiti proizvoljno puta, svaki put s novim brojčanim podacima. Ova mogućnost se pri programiranju redovito koristi, pa se program sastavlja tako da se potrebni brojčani podaci daju računalu tek u drugoj fazi rada (izvršenje) a ne kao unutar prvog poglavlja već u prvoj fazi rada (učitavanje programa i kompiliranje).

Zbog takve organizacije rada potrebno je računalu nakon naredbe za izvršenje programa dati odgovarajuće brojčane podatke, koji se obično nazivaju ulazni podaci, ili skraćeno podaci. Za razliku od brojčanih vrijednosti, koje su kao konstante ugrađene u pojedine naredbe u programu, kod ulaznih podataka daju se samo čiste brojčane vrijednosti i to (kao što je već rečeno) tek nakon naredbe za izvršenje. Za ovaj program dat ćemo ulazne podatke na slijedeći način:

```
   12.1234    121234 12.123456
     1234         12   123456.
```

Rezultat izvršenja programa u ovoj lekciji, uključivo učitavanje ulaznih podataka, bit će na izlaznoj listi slijedećeg oblika.

```
  12.1234   12.1234   12.1234
   0.1234    0.0012**********
```

Ulazni podaci (za razliku od konstanata u programu) unose se u računalo tek prilikom izvršenja programa, analogno kao što se tek tada dobivaju i otisnuti izlazni rezultati.

Pisanje ulaznih podataka vrši se na formularu za 80 znakova u jednom retku. Svaki redak bit će kasnije prenesen na jednu bušenu karticu. Za pisanje ulaznih podataka može se koristiti i FORTRAN-formular.

Kartica (bušena kartica) je sredstvo za unošenje ulaznih podataka u računalo za vrijeme izvršenja programa. To se obavlja pomoću čitala kartica.

Čitalo kartica je ulazna jedinica, putem koje se potrebne informacije unose u računalo. Karakteristika ove ulazne jedinice u FORTRAN-u jest brojka 2.

READ (......., ...) .... je izvršna naredba za čitanje ulaznih podataka i pridavanje tih brojčanih vrijednosti pojedinim varijablama. Prva brojka u zagradi je karakteristika ulazne jedinice, a druga se odnosi na pripadni format. Iza zagrade nalaze se varijable, kojima se pridaju učitane brojčane vrijednosti.

Obrnuti trapez u shemi toka je simbol za učitavanje podataka.

F-format se za čitanje ulaznih podataka upotrebljava analogno kao za tiskanje izlaznih rezultata.

Decimalna točka kod realnih ulaznih podataka se može izostaviti. U tom slučaju računalo uzima broj decimalnih mjesta prema specifikaciji u naredbi FORMAT (implicitno određivanje broja decimalnih mjesta).

Decimalna točka kod realnih ulaznih podataka može se pisati tako da se to slaže sa specifikacijom u naredbi FORMAT (eksplicitno i implicitno određivanje broja decimalni mjesta su međusobno u skladu).

Decimalna točka kod realnih ulaznih podataka može se pisati tako da se to ne slaže sa specifikacijom u naredbi FORMAT. U tom slučaju računalo

uzima broj decimalnih mjesta kako je to napisano kod ulaznih podataka (eksplicitno određivanje broja decimalnih mjesta).

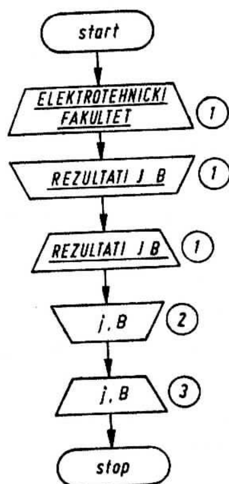
Ista naredba FORMAT može se povezati s više ulazno-izlaznih naredaba, tako da se u većem broju naredaba READ, odnosno WRITE, može nalaziti isti broj za naredbu FORMAT.

21. LEKCIJA

UČITAVANJE TEKSTA I PRESKOK

```
C         STAMPANJE NAPISANOG TEKSTA
      WRITE(3,1)
    1 FORMAT(25H ELEKTROTEHNICKI FAKULTET)
C         UCITAVANJE NOVOG TEKSTA
      READ(2,1)
C         STAMPANJE UCITANOG TEKSTA
      WRITE(3,1)
C         PRESKOK KOD CITANJA
      READ(2,2)J,B
    2 FORMAT(5X,I5,10X,F10.0)
C         PRESKOK KOD STAMPANJA
      WRITE(3,3)J,B
    3 FORMAT(10X,I5,F10.2)
      CALL EXIT
      END
```

Zadatak je u ovoj lekciji da se tekst i brojčane vrijednosti učitaju i otisnu. Shema toka je veoma jednostavna i matematičkih operacija uopće nema.



Svrha je, međutim, ove lekcije da naučimo kako možemo neki tekst učitati i onda ga po potrebi otisnuti. Time je omogućeno tiskanje teksta koji nije fiksiran u programu, nego ga u računalo unosimo za vrijeme izvršenja. Time se postiže općenitost programa i u pogledu tiskanja tekstova.

Osim toga, uvest ćemo ovdje i preskok kod učitavanja i tiskanja. Pri tome se kod učitavanja preskače određeni broj kućica, bez obzira postoji li tu neka informacija ili ne, i ta se informacija ne unosi u računalo. Pri tiskanju također se preskače određeni broj kućica i tamo onda ostaju bjeline.

Ulazni podaci u ovoj lekciji imaju slijedeći oblik. Oni, dakako, sadrže i tekst (prvi redak), a ne samo brojčane podatke (drugi redak).

```
 REZULTATI     J         B
   24   25     17.84     25.43
```

Rezultat rada računala nakon izvršenja ovog programa imat će slijedeći oblik.

```
ELEKTROTEHNICKI FAKULTET
REZULTATI     J         B
             25     25.43
```

H-format za učitavanje teksta može se upotrijebiti tako da se naredba READ pozove na naredbu FORMAT, gdje se već nalazi neki proizvoljni tekst iza specifikacije za H-format.

Učitavanje teksta u H-formatu dovodi do toga da se od ulaznih podataka učita toliko znakova koliko je specificirano H-formatom, i to smjesti tamo gdje se prije nalazio proizvoljni tekst (novi tekst prekriva ono što je tamo bilo prije).

X-format uz naredbu READ kaže koliko kućica u ulaznim podacima treba pri čitanju preskočiti, bez obzira da li se tu nalaze neki podaci ili ne.

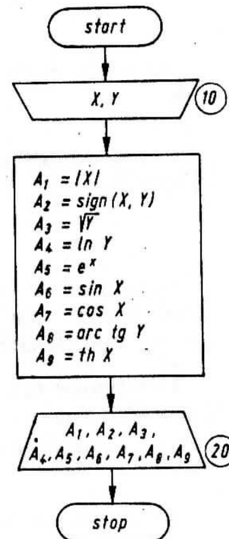
X-format uz naredbu WRITE kaže koliko kućica pri tiskanju izlaznih informacija treba preskočiti, pa na tom mjestu neće biti otisnuto ništa, već ostaju bjeline.

Miješani format sadrži specifikacije različitog tipa, npr. za cijele brojeve, realne brojeve ili tekst. Pri tome specifikacije moraju biti usklađene po redoslijedu, npr. kad dođe na red učitavanje po formatu I5, onda se to mora odnositi na cjelobrojnu varijablu naredbi READ, a slijedećih 5 kućica u ulaznim podacima mora sadržavati cijeli broj (a ne npr. 25.43 ili SPLIT).

22. LEKCIJA

REALNE KOMPILATORSKE FUNKCIJE

```
C          UCITAVANJE ARGUMENATA
      READ(2,10)X,Y
C          REALNE KOMPAJLERSKE FUNKCIJE
      A1=ABS(X)
      A2=SIGN(X,Y)
      A3=SQRT(Y)
      A4=ALOG(Y)
      A5=EXP(X)
      A6=SIN(X)
      A7=COS(X)
      A8=ATAN(Y)
      A9=TANH(X)
C          STAMPANJE
      WRITE(3,20)A1,A2,A3,A4,A5,A6,A7,A8,A9
   10 FORMAT(5F10.0)
   20 FORMAT(5F10.4)
      CALL EXIT
      END
```



Zadatak u ovoj lekciji jest da se učitaju dvije vrijednosti, koje će biti upotrijebljene kao argumenti pri izračunavanju vrijednosti različitih funkcija. Funkcije koje ćemo izračunati neka budu sve raspoložive realne kompilatorske funkcije. Na kraju treba otisnuti dobivene rezultate. Svrha je da se upoznamo s kompilatorskim funkcijama, koje imaju realnu vrijednost. Treba napomenuti da kod ovih funkcija i argumenti imaju realnu vrijednost.

Algoritme za ovu lekciju nećemo postavljati, jer su funkcije definirane u samom FORTRAN-kompilatoru. Pri korištenju ovih funkcija nije niti potrebno da znamo kako izgledaju algoritmi za njihovo izračunavanje. Budući da ne sadrži algoritme, shema toka je veoma jednostavna.

Ulazni podaci sadrže samo dvije realne vrijednosti.

```
      -0.5      4.00
```


Rezultat izvršenja programa imat će ovaj oblik:

```
    0.5000    0.5000    2.0000    1.3862    0.6065
   -0.4794    0.8775    1.3258   -0.4621
```

Realna kompilatorska funkcija definirana je u FORTRAN-kompilatoru. Kod tih funkcija i vrijednosti funkcija i vrijednosti argumenata su realne.

Ime kompilatorske funkcije određeno je u FORTRAN-kompilatoru. Građeno je na isti način kao i imena varijabli, a početnim slovom implicitno je određeno da je to realna funkcija.

ABS je kompilatorska funkcija za izračunavanje apsolutne vrijednosti realnog argumenta.

SIGN je kompilatorska funkcija čiji rezultat je apsolutna vrijednost prvog argumenta s predznakom drugog argumenta (oba argumenta su realna).

SQRT je kompilatorska funkcija za izračunavanje drugog korijena od realnog argumenta.

ALOG je kompilatorska funkcija za izračunavanje prirodnog logaritma. Kao prvo slovo stavljeno je A, jer se kao rezultat dobiva realna vrijednost.

EXP je kompilatorska funkcija za izračunavanje vrijednosti baze prirodnih logaritama (2.718281) na potenciju argumenta.

SIN je kompilatorska funkcija za izračunavanje sinusa. Vrijednost argumenta uzima se u radijanima.

COS je kompilatorska funkcija za izračunavanje kosinusa. Vrijednost argumenta uzima se u radijanima.

ATAN je kompilatorska funkcija za izračunavanje arkustangensa. Rezultat se dobiva u radijanima, i to unutar prvog i četvrtog kvadranta.

TANH je kompilatorska funkcija za izračunavanje hiperbolnog tangensa.

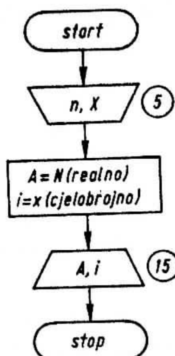
23. LEKCIJA

KOMBINIRANE KOMPILATORSKE FUNKCIJE

```
C      UCITAVANJE ARGUMENATA
       READ(2,5)N,X
     5 FORMAT(I10,F10.0)
C      REALNA VRIJEDNOST
       A=FLOAT(N)
C      CJELOBROJNA VRIJEDNOST
       I=IFIX(X)
C      STAMPANJE
       WRITE(3,15)A,I
    15 FORMAT(F10.4,I10)
       CALL EXIT
       END
```

Zadatak u ovoj lekciji jest da se učitaju jedna cjelobrojna i jedna realna vrijednost, a zatim da se cjelobrojna pretvori u realnu, a realna u cjelobrojnu. Pri tome koristimo funkcije koje su također već sadržane u FORTRAN--kompilatoru. Kod ovih funkcija vrijednost funkcije i vrijednost argumenata nisu istog tipa (jedna je realna, a druga cjelobrojna).

Algoritmi su sadržani u kompilatoru, pa ih ne treba postavljati, a shema toka je posve jednostavna.



Ulazni podaci sadrže jednu cjelobrojnu i jednu realnu vrijednost.

```
         5      7.123
```

Rezultat izvršenja programa bit će ovog oblika:

```
    5.0000         7
```

Kombinirana kompilatorska funkcija ima vrijednost funkcije jednog tipa, a vrijednost argumenta drugog tipa. Na primjer, vrijednost funkcije je realna, a vrijednost argumenta cjelobrojna. Ili obrnuto.

FLOAT je kompilatorska funkcija za dobivanje realne vrijednosti od cjelobrojnog argumenta. Ime potječe od »floating point« što se svojedobno upotrebljavalo za označavanje realnih vrijednosti.

IFIX je kompilatorska funkcija za dobivanje cjelobrojne vrijednosti od realnog argumenta. Ime potječe od »fixed point«, što se svojedobno upotrebljavalo za označavanje cjelobrojnih vrijednosti (slovo I je dodano na početku, jer je to cjelobrojna vrijednost).

Upotreba kombiniranih funkcija je danas razmjerno rijetka, jer FORTRAN dopušta upotrebu veličine različitog tipa u istoj aritmetičkoj naredbi. Ipak ćemo ih u izvjesnim slučajevima upotrijebiti. U ovom programu mogli smo ih izbjeći, ali su namjerno upotrijebljene zbog ilustracije.

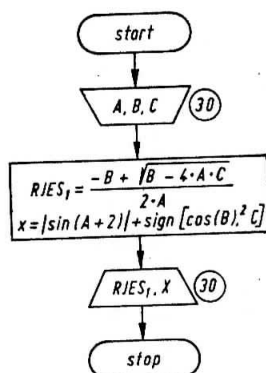
24. LEKCIJA

POZIVANJE KOMPILATORSKIH FUNKCIJA

```
C        UCITAVANJE PODATAKA
      READ(2,30)A,B,C
C        POZIVANJE FUNKCIJA
      RJES1=(-B+SQRT(B**2-4.0*A*C))/(2.0*A)
      X=ABS(SIN(A+2.))+SIGN(COS(B**2),C)
C        STAMPANJE REZULTATA
      WRITE(3,30)RJES1,X
   30 FORMAT(3F10.4)
      CALL EXIT
      END
```

Zadatak je ove lekcije da se riješe dva aritmetička izraza u kojima se koriste kompilatorske funkcije. U izrazima se nalaze neke veličine čiju brojčanu vrijednost treba najprije učitati. Prvi izraz odgovara jednom rješenju kvadratne jednadžbe.

Nikakav složeni algoritam ovdje nije potreban, jer su traženi rezultati u aritmetičkim izrazima dani eksplicitno. Shema toka, kao najvažniji dio, sadrži upravo te izraze.



Ulazni podaci sadrže tri realne vrijednosti.

```
      2.0     -3.0     -9.0
```

Dobiva se slijedeći rezultat izvršenja programa:

```
    3.0000   -0.1543
```

Pozivanje kompilatorske funkcije vrši se tako da se u aritmetičkom izrazu (s desne strane znaka jednakosti) na odgovarajućem mjestu napiše ime funkcije, a iza toga u zagradi se napiše argument (odnosno dva argumenta kod funkcija SIGN i ISIGN).

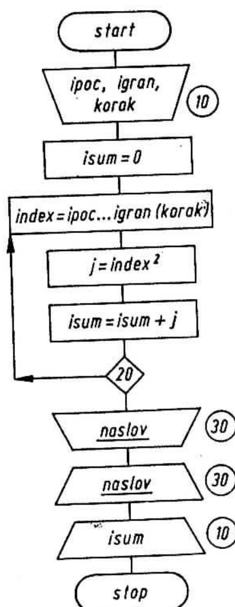
Argument (u zagradi iza imena funkcije) može biti napisan kao bilo kakav aritmetički izraz. To uključuje da argument bude samo jedna varijabla, ili samo jedna konstanta.

Funkcija unutar argumenata može se upotrijebiti kao aritmetički operand bez ikakvih ograničenja. Funkcija sa svojim argumentima nalazi se tada unutar izraza za argument neke druge funkcije.

25. LEKCIJA

PETLJA

```
C        RUTINE ISPRED PETLJE
      READ(2,10)IPOC,IGRAN,KORAK
   10 FORMAT(3I10)
      ISUM=0
C        POCETAK PETLJE
      DO 20 INDEX=IPOC,IGRAN,KORAK
      J=INDEX**2
   20 ISUM=ISUM+J
C        IZLAZ IZ PETLJE
      READ(2,30)
   30 FORMAT('REZERVIRANO ZA NASLOV REZULTATA                 ')
      WRITE(3,30)
      WRITE(3,10)ISUM
      CALL EXIT
      END
```



Zadatak je ove lekcije da se izračuna suma kvadrata neparnih brojeva od 1 do 50. Program je načinjen općenito, tako da se može izračunati suma kvadrata bilo kojeg niza brojeva s konstantnom diferencijom. Da su u ovoj lekciji to upravo neparni brojevi do 50, određeno je podacima koje računalo učita prilikom izvršenja programa. Svrha lekcije jest da se uvede petlja, koja se veoma često upotrebljava u programima za ponavljanje neke rutine određeni broj puta.

Algoritam je tipičan za izračunavanje sume. Prije početka petlje postavi se vrijednost varijable, predviđene za sumu, jednaka nuli. U svakoj petlji dodaje se staroj vrijednosti te varijable daljnji član. Nakon izvršenja čitavog ponavljanja petlje dobiva se konačna suma, koja se može dalje koristiti nakon što se izađe iz petlje.

Ulazni podaci sadržavaju tri brojčana podatka, koji definiraju zadani niz, a osim toga tekst naslova, koji će biti otisnut prije rezultata.

```
         1         50        2
SUMA KVADRATA NEPARNIH BROJEVA DO 50 JE
```

Rezultat izvršenja programa dobiva se u ovom obliku:

```
SUMA KVADRATA NEPARNIH BROJEVA DO 50 JE
     20825
```

DO je izvršna naredba za ponavljanje neke rutine određeni broj puta. Ta naredba je početak petlje koja će se ponavljati. Iza DO u istoj naredbi mora biti naveden broj završne naredbe, a zatim mora biti definiran indeks petlje.

Broj naredbe iza DO definira završnu naredbu, tj. naredbu koja čini završetak petlje. Završna naredba uključena je u petlji, te se izvršava na kraju svakog ponavljanja petlje.

Završna naredba implicitno sadrži i uputu za skok na početak petlje, sve dok se petlja ne izvrši određeni broj puta. Tek nakon toga program se nastavlja slijedećom naredbom.

Indeks petlje je cjelobrojna varijabla, a nalazi se u naredbi DO iza broja završne naredbe. Prilikom ponavljanja petlje mijenja se vrijednost indeksa automatski, ali unutar svakog izvršenja petlje vrijednost indeksa ostaje nepromijenjena.

Početna vrijednost indeksa mora biti veća od nule. Definirana je prvom cjelobrojnom vrijednošću iza znaka jednakosti. U tu svrhu može se upotrijebiti cjelobrojna varijabla ili cjelobrojna konstanta.

Granična vrijednost indeksa redovito je veća od početne vrijednosti. Definirana je drugom cjelobrojnom vrijednošću iza znaka jednakosti. U tu svrhu može se upotrijebiti cjelobrojna varijabla ili cjelobrojna konstanta. Petlja se ponavlja sve dotle dok je indeks manji od granične vrijednosti ili joj je jednak.

Korak indeksa jest povećanje vrijednosti indeksa za svako ponavljanje petlje. Korak mora biti pozitivan. Definiran je trećom cjelobrojnom vrijednošću iza znaka jednakosti. To je eksplicitni način definiranja koraka. U tu svrhu može se upotrijebiti cjelobrojna varijabla ili cjelobrojna konstanta.

Upotreba indeksa za računanje unutar petlje je dopuštena, pa se indeks smije nalaziti u aritmetičkom izrazu (s desne strane znaka jednakosti). Ali vrijednost indeksa ne smijemo namjerno mijenjati, pa se indeks ne smije nalaziti u aritmetičkoj naredbi s lijeve strane znaka jednakosti.

Izlaz iz petlje nastupa kad vrijednost indeksa postane veća od definirane granične vrijednosti. Tada se petlja više ne ponavlja, nego se izvršava slijedeća naredba iza završne naredbe petlje. Ako indeks prilikom ponavljanja petlje postigne točno graničnu vrijednost, onda će se petlja još ponoviti (posljednji put) s tom vrijednošću indeksa.

Pravokutnik u shemi toka s upisanim indeksom, početnom i graničnom vrijednošću, te korakom koristi se za označavanje početka petlje.

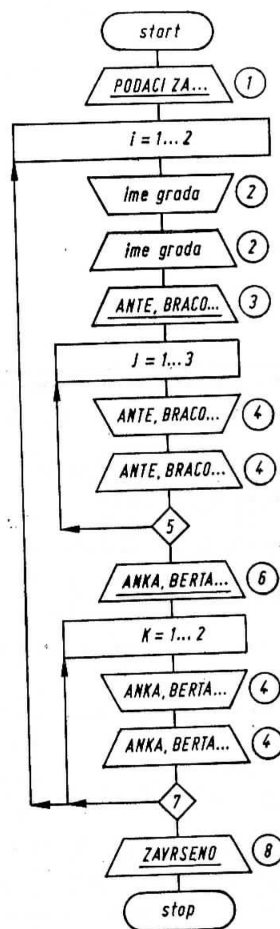
Uspravni kvadrat u shemi toka, s upisanim brojem završne naredbe, koristi se za označavanje završetka petlje.

LEKCIJA **26.**

PETLJA U PETLJI

```
C         STAMPANJE NASLOVA
      WRITE(3,1)
 1    FORMAT(' PODACI ZA RAZNE GRADOVE')
C         POCETAK ZAJEDNICKE PETLJE
      DO 7 I=1,2
      READ(2,2)
 2    FORMAT('REZERVIRANO ZA IME GRADA')
      WRITE(3,2)
      WRITE(3,3)
 3    FORMAT('      ANTE      BRACO     IVAN      JANKO')
C         POCETAK PRVE PETLJE
      DO 5 J=1,3
      READ(2,4)ANTE,BRACO,IVAN,JANKO
 4    FORMAT(2F10.3,2I10)
 5    WRITE(3,4)ANTE,BRACO,IVAN,JANKO
C         IZLAZ IZ PRVE PETLJE
      WRITE(3,6)
 6    FORMAT('      ANKA      BERTA     IVKA      JULKA')
C         POCETAK DRUGE PETLJE
      DO 7 K=1,2
      READ(2,4)ANKA,BERTA,IVKA,JULKA
 7    WRITE(3,4)ANKA,BERTA,IVKA,JULKA
C         IZLAZ IZ ZAJEDNICKE I IZ DRUGE PETLJE
      WRITE(3,8)
 8    FORMAT(' ZAVRSENO')
      CALL EXIT
      END
```

Zadatak u ovoj lekciji jest da se učita niz podataka, i to opet otisne u obliku tablice. Pri tome se neke radnje (učitavanje, tiskanje) ponavljaju, a unutar toga se druge analogne radnje također ponavljaju. To dovodi do toga da u programu imamo jednu petlju unutar druge petlje. Zadatak nećemo posebno opisivati, jer je vidljiv iz sheme toka.



Ulazni podaci neka izgledaju ovako:

```
ZAGREB
     1.111      11.111          11        111
     2.222      22.222          22        222
     3.333      33.333          33        333
     4.444      44.444          44        444
     5.555      55.555          55        555
OSIJEK
     5.555      55.555          55        555
     6.666      66.666          66        666
     7.777      77.777          77        777
     8.888      88.888          88        888
     9.999      99.999          99        999
```


Rezultat izvršenja programa bit će ovog oblika:

```
PODACI ZA RAZNE GRADOVE
ZAGREB
      ANTE       BRACO         IVAN       JANKO
     1.111      11.111           11         111
     2.222      22.222           22         222
     3.333      33.333           33         333
      ANKA       BERTA         IVKA       JULKA
     4.444      44.444           44         444
     5.555      55.555           55         555
OSIJEK
      ANTE       BRACO         IVAN       JANKO
     5.555      55.555           55         555
     6.666      66.666           66         666
     7.777      77.777           77         777
      ANKA       BERTA         IVKA       JULKA
     8.888      88.888           88         888
     9.999      99.999           99         999
ZAVRSENO
```

Petlja unutar petlje smije se nalaziti, ali se vanjska i unutarnja petlja ne smiju preklapati (*»petlje se ne smiju zapetljati«).*

Više petlja u petlji smije se nalaziti, i unutarnje petlje mogu biti bilo jedna iza druge, bilo jedna unutar druge.

Zajednički završetak petlji smije se koristiti, tako da je ista naredba završetak za vanjsku i za unutarnju petlju.

Redoslijed izvršavanja petlji je uvijek takav da se prije izvrše sva ponavljanja unutarnje petlje, a tek zatim se ponovi vanjska petlja. O ovome je posebno važno voditi računa kod zajedničkog završetka vanjske i unutarnje petlje.

Konstanta se može koristiti za definiranje početne i granične vrijednosti, te za korak indeksa.

Korak indeksa može biti definiran implicitno, ali samo ako iznosi +1. U tom slučaju vrijednost koraka se ne mora navesti u naredbi DO.

27. LEKCIJA

PETLJA I SKOKOVI

```
C        KOLIKO BROJEVA JE DJELJIVO SA 7
      KOLIK=0
C        UCITAJ I OBRADI 100 CIJELIH BROJEVA
      DO 60 I=1,100
      READ(2,10)N
   10 FORMAT(I10)
C        JE LI BROJ NEGATIVAN
      IF(N)20,40,40
C        AKO JEST, JAVI I PREKINI RAD
   20 WRITE(3,30)I
   30 FORMAT(I4,'-TI BROJ JE NEGATIVAN')
      GO TO 80
C        AKO NIJE, JE LI DJELJIV SA 7
   40 IF(N-N/7*7)50,50,60
C        AKO JEST DJELJIV, UBROJI GA
   50 KOLIK=KOLIK+1
C        AKO NIJE, NASTAVI S CITANJEM
   60 CONTINUE
C        ISPISI REZULTAT
      WRITE(3,70)KOLIK
   70 FORMAT(' OD 100 BROJEVA' I4 ' JE DJELJIVO SA 7')
   80 CALL EXIT
      END
```

Zadatak je ove lekcije da se učita 100 zadanih brojeva i ispita koliko njih je djeljivo sa 7, pa da se to ispiše kao rezultat. Ukoliko se zabunom među podacima nađe negativan broj, računalo treba da dade obavijest o tome i da prekine rad. Svrha je ove lekcije da se uvedu skokovi u vezi s petljom.

Shema toka pokazuje postupak. Rutinu učitavanja i ispitivanja djeljivosti treba ponoviti 100 puta. Ako učitani broj jest djeljiv sa 7, treba ga pribrojiti u konačnu sumu, inače ne. Na taj način dolazi do uvjetnog preskakivanja jednog dijela rutine unutar petlje; ako je učitani broj negativan, treba iskočiti iz petlje i prekinuti ponavljanje.

start
kolik = 0
i = 1 ... 100
n (10)
n
< 0 (20)
≥ 0 (40)
i-TI BROJ JE NEGATIVAN (30)
n djeljivo sa 7
NE
DA (50)
kolik = kolik + 1
(60)
OD 100 BROJEVA kolik JE DJELJIVO (70)
(80)
stop

U ovoj lekciji ćemo prikazati dva izvršenja programa.

U prvom izvršenju programa neka se među zadanim podacima nalazi i jedan negativan broj.

```
    5
    6
    7
    8
   -9
i tako dalje
```

Rezultat izvršenja programa bit će ovakav:

```
   5-TI BROJ JE NEGATIVAN
```

U drugom izvršenju programa neka se među podacima nalaze samo 3 koja su djeljiva sa 7.

```
7
7
7
8
8
i tako dalje same osmica
```

Rezultat rada bit će ovakav:

```
OD 100 BROJEVA   3 JE DJELJIVO SA 7
```

Skok unutar petlje je dopušten. Skok može biti prema naprijed ili prema natrag. To može biti uvjetan ili bezuvjetan skok.

Skok iz petlje prema van je dopušten. U tom slučaju indeks zadržava svoju vrijednost i može se koristiti. (Skok izvana u petlju nije dopušten.)

CONTINUE je izvršna naredba, koja znači samo to da treba nastaviti s radom. Ona je implicitno sadržana u svim izvršnim naredbama (jer nakon izvršenja te naredbe treba izvršiti slijedeću), osim u saobraćajnim naredbama (koje eksplicitno znače da se ne nastavlja sa slijedećom naredbom, nego s nečim drugim).

Upotreba naredbe CONTINUE je najčešća na kraju petlje, jer se često bez nje ne može završiti petlja.

Skok na početak petlje ne smije se izvršiti direktno, jer u tom slučaju neće biti ispitano da li je petlja izvršena dovoljni broj puta, već se mora izvršiti skokom na završetak petlje. Ako pri tome nije potrebno izvršiti nikakvu operaciju, završna naredba mora biti CONTINUE.

Saobraćajna naredba na kraju petlje ne može preuzeti ulogu završne naredbe. Zato se u takvom slučaju dodaje naredba CONTINUE, koja tu ulogu preuzima.

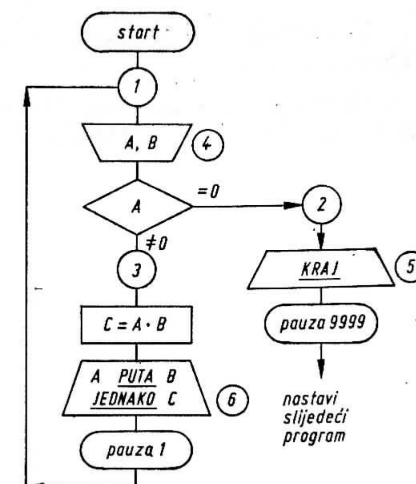
28. LEKCIJA

PRIVREMENO ZAUSTAVLJANJE

```
C        POCETAK RUTINE
    1 READ(2,4)A,B
      IF(A)3,2,3
C        ZAUSTAVLJANJE NA ZAVRSETKU RADA
    2 WRITE(3,5)
      STOP 9999
C        IZVRSENJE RUTINE
    3 C=A*B
      WRITE(3,6)A,B,C
         ZAUSTAVLJANJE PRIJE PONAVLJANJA
      PAUSE 1
      GO TO 1
    4 FORMAT(2F10.0)
    5 FORMAT(' KRAJ')
    6 FORMAT(F10.4 ' PUTA' F10.4 ' =' F10.4)
      END
```

Zadatak ove lekcije jest da se pomnože dva broja i to ponavlja dok god ima podataka. Svrha je da se uvede privremeno zaustavljanje, jer je program sastavljen tako da se računalo zaustavi svaki put kad izvrši jedno množenje. Isto tako u programu je zadano i privremeno zaustavljanje prije prijelaza na izvršenje slijedećeg programa.

U shemi toka umetnut je i uvjetan skok na završetku rada, ako nema više podataka. To se postiže praznom karticom na kraju programa, što računalo učita kao vrijednost nula.



Ulazni podaci sadržavaju po dva podatka na svakoj kartici. To su vrijednosti koje treba množiti.

```
         3         4
         5         6
   prazna kartica
```

Na kraju se nalazi jedna prazna kartica.

Rezultat izvršenja programa izgleda ovako. Pri tome će se računalo zaustaviti nakon tiskanja svakog retka. Računalo će svaki put nastaviti rad nakon što operater ručno pritisne odgovarajuće tipkalo.

```
   3.0000 PUTA     4.0000 =    12.0000
   5.0000 PUTA     6.0000 =    30.0000
KRAJ
```

STOP je saobraćajna naredba koja kod računala s monitorom znači privremeno zaustavljanje na kraju izvršenja programa. Pritiskom na tipkalo »PROGRAM START« rad se nastavlja, ali ne s istim programom, nego monitor preuzima slijedeći JOB.

Broj uz STOP može se upotrijebiti (ali ne mora) kao obavijest operateru, i taj se broj pojavljuje na signalnoj ploči računala prilikom zaustavljanja. Smije se upotrijebiti bilo koji pozitivni cijeli broj do uključivo 9999.

PAUSE je saobraćajna naredba za privremeno zaustavljanje računala unutar izvršenja istog programa. Pritiskom na tipkalo »PROGRAM START« izvršenje programa se nastavlja i to slijedećom naredbom.

Broj uz PAUSE može se upotrijebiti isto kao uz naredbu STOP.

Učitavanje bjelina kao brojčanih podataka računalo interpretira kao vrijednosti nula.

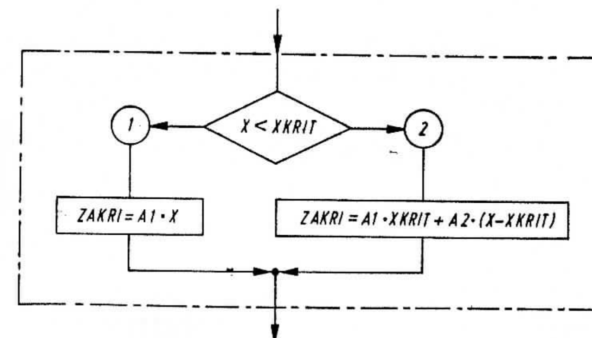
Prazna kartica na kraju ulaznih podataka može se koristiti za završetak rutine uz odgovarajuću naredbu IF.

LEKCIJA **29.**

FUNKCIJE

```
C        APROKSIMACIJA KRIVULJE ZASICENJA
      FUNCTION ZAKRI (A1,XKRIT,A2,X)
C        A1   =NAGIB STRMOG USPONA
C        XKRIT=GRANICA STRMOG I BLAGOG USPONA
C        A2   =NAGIB BLAGOG USPONA
C        X    =VRIJEDNOST APSCISE
      B=X-XKRIT
      IF(B)1,2,2
  1   ZAKRI=A1*X
      RETURN
  2   ZAKRI=A1*XKRIT+A2*B
      RETURN
      END
```

Zadatak ove lekcije jest da se postavi potprogram za računanje ordinate u krivulji zasićenja, koja krivulja je aproksimirana sa dva pravca. Prvi pravac polazi iz ishodišta i strmiji je. Od određene vrijednosti apscise nastavlja se drugi pravac s blažim usponom. Već prema veličini apscise, treba izabrati odgovarajući izraz za računanje ordinate. Potprogram je sastavljen općenito za bilo koje vrijednosti nagiba i za bilo koju granicu nagiba. Svrha ove lekcije jest da se uvedu potprogrami u obliku funkcija. Prikazani potprogram jest definicija takve funkcije.



Shema toka za definiciju ove funkcije nema početka ni završetka, jer je to potprogram koji se može izvršiti samo u okviru nekog (glavnog) programa. Ulaznih podataka niti rezultata izvršenja također nema u ovoj lekciji zbog istih razloga. U ovoj lekciji smo samo definirali jednu funkciju i napisali odgovarajući potprogram.

Potprogram je rutina koja se izvršava (jedan ili više puta) u okviru nekog glavnog programa.

Glavni program je program u kojem se (između ostalih naredbi) jedan ili više potprograma poziva na izvršenje.

Funkcija je potprogram čija definicija zahtijeva više naredbi, ali se dobiva samo jedan rezultat.

FUNCTION je organizaciona naredba koja sadrži ime funkcije i argumente, i kojom se započinje definicija potprograma u obliku funkcije.

Definicija funkcije počinje naredbom FUNCTION i završava sa END, te sadrži sve naredbe koje služe za računanje vrijednosti funkcije. Definicija je samo organizacionog karaktera, kako se ima taj potprogram izvršiti kad bude pozvan u glavnom programu.

Ime funkcije mora se nalaziti iza FUNCTION, a gradi se isto kao imena varijabli. Tip funkcije (realna ili cjelobrojna) određen je prvim slovom imena.

Argumenti se u naredbi FUNCTION moraju nalaziti u zagradi iza imena funkcije. U definiciji su to samo formalni argumenti, koji služe da se definira algoritam po kojem će se vršiti računanje sa vrijednostima stvarnih argumenata navedenih u pozivu funkcije.

Imena argumenata u definiciji funkcije neovisna su o imenima izvan te definicije. Grade se isto kao što je rečeno za imena varijabli.

Vrijednost funkcije mora biti izračunata barem jedanput unutar definicije (ime funkcije mora se nalaziti s lijeve strane znaka jednakosti).

Imena varijabli upotrijebljena u definiciji funkcije neovisna su o imenima izvan te definicije.

Brojevi naredbi upotrijebljeni u definiciji funkcije neovisni su o brojevima naredbi izvan te definicije.

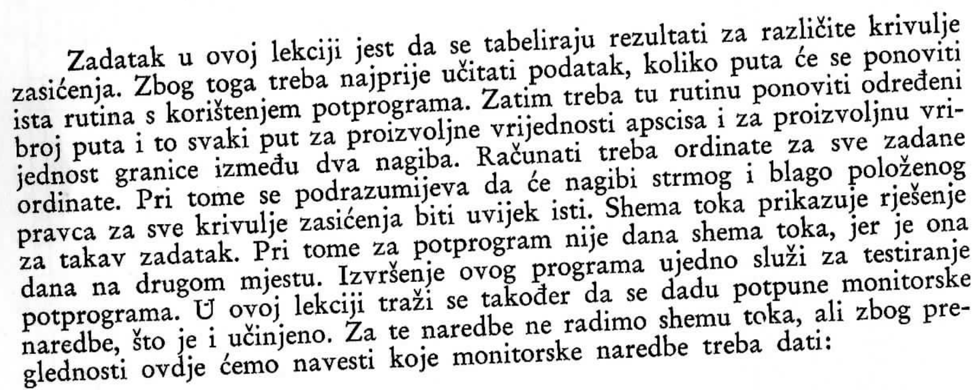
RETURN je saobraćajna naredba koja određuje kada će se pri izvršenju potprograma skočiti natrag u glavni program. U potprogramu mora biti barem jedna naredba RETURN.

Pravokutnik crta-točka upotrebljava se u shemi toka za uokvirivanje potprograma.

LEKCIJA **30.**

MONITOR I FUNKCIJE

```
// JOB T          STEFANiNi
// FOR
*LiST SOURCE PROGRAM
C         POTPROGRAM ZA KRiVULJU ZASiCENJA
      FUNCTiON ZAKRi(A1,XKRiT,A2,X)
      B=X-XKRiT
      iF(B)1,2,2
    1 ZAKRi=A1*X
      RETURN
    2 ZAKRi=A1*XKRiT+A2*B
      RETURN
      END
// DUP
*STORE      WS  UA  ZAKRi
// FOR
*iOCS(CARD,1132PRiNTER)
*LiST SOURCE PROGRAM
C         GLAVNi PROGRAM
C       UVODNi DiO
      READ(2,1)i1
    1 FORMAT(i1Ø)
      DO 5 i=1,i1
      READ(2,2)XØ,J1,J2,J3
    2 FORMAT(F1Ø.0,3i1Ø)
      WRiTE(3,3)
    3 FORMAT('    APSCiSA  ORDiNATA')
C     . POZiVANJE FUNKCiJE i STAMPANJE
      DO 5 J=J1,J2,J3
      ORDi=ZAKRi(5.,XØ,Ø.5,FLOAT(J))
      WRiTE(3,4)J,ORDi
    4 FORMAT(i1Ø,F10.2)
    5 CONTiNUE
      CALL EXiT
      END
// XEQ
```



Zadatak u ovoj lekciji jest da se tabeliraju rezultati za različite krivulje zasićenja. Zbog toga treba najprije učitati podatak, koliko puta će se ponoviti ista rutina s korištenjem potprograma. Zatim treba tu rutinu ponoviti određeni broj puta i to svaki put za proizvoljne vrijednosti apscisa i za proizvoljnu vrijednost granice između dva nagiba. Računati treba ordinate za sve zadane ordinate. Pri tome se podrazumijeva da će nagibi strmog i blago položenog pravca za sve krivulje zasićenja biti uvijek isti. Shema toka prikazuje rješenje za takav zadatak. Pri tome za potprogram nije dana shema toka, jer je ona dana na drugom mjestu. Izvršenje ovog programa ujedno služi za testiranje potprograma. U ovoj lekciji traži se također da se dadu potpune monitorske naredbe, što je i učinjeno. Za te naredbe ne radimo shemu toka, ali zbog preglednosti ovdje ćemo navesti koje monitorske naredbe treba dati:

1. Izvrši slijedeći posao kao privremen.
2. Koristi FORTRAN-kompilator i kompiliraj slijedeći program iz FORTRANA u jezik računala.
2.1 Pri tome otisni listu izvornog programa.

Ovdje dolazi potprogram.

3. Koristi pomoćne rutine na disku, i to
3.1 Ono što je upravo kompilirano spremi na disk pod imenom ZAKRI.
4. Koristi FORTRAN-kompilator (kao gore).
4.1 Pri tome su potrebne rutine za čitalo kartica i za tiskaljku.
4.2 Pri tome otisni listu izvornog programa.

Ovdje dolazi glavni program.

5. Izvrši program koji je upravo kompiliran.

Sve ovo što je gore navedeno, napisano je na FORTRAN-formularu u obliku odgovarajućih naredbi. To se prenese na bušene kartice i stavi u čitalo kartica, a zatim se pritiskom na tipkalo za start pokrene računalo. Ulazni podaci izbuše se na kartice koje slijede u čitalu kartica odmah iza prije navedenoga. U našoj lekciji ulazni podaci izgledaju ovako:

```
         1
        5.         1        10         1
```

Rezultat izvršenja programa u ovom slučaju izgledat će ovako (ovdje su navedeni samo rezultati, a izostavljeno je sve ostalo što računalo prije toga tiska, tj. lista naredaba i obavijesti u vezi sa spremanjem na disk i kompiliranjem):

APSCISA	ORDINATA
1	5.00
2	10.00
3	15.00
4	20.00
5	25.00
6	25.50
7	26.00
8	26.50
9	27.00
10	27.50

// **DUP** je monitorska naredba kojom se pozivaju pomoćne rutine s diska. Nakon ove naredbe dolazi upravljačka naredba koja kaže koju pomoćnu rutinu želimo koristiti i kako.

* **STORE** je DUP — upravljačka naredba za spremanje programa (odnosno potprograma) na disk. U toj naredbi mora biti označeno odakle se sprema, zatim područje diska kamo se sprema i konačno pod kojim imenom se to sprema.

WS je kratica za radno područje na disku (Working Storage). Tu je smješten svaki program neposredno nakon kompiliranja, ali ne za trajno, jer će ga prekriti slijedeći kompilirani program.

UA je kratica za korisnikovo područje na disku (User's Area). Tu se pomoću naredbe *STORE može program spremiti za trajniju upotrebu.

***IOCS (CARD, 1132 PRINTER)** je FORTRAN-upravljačka naredba za povezivanje rutina potrebnih čitalu kartica i linijskoj tiskaljci. Ova naredba stavlja se samo ispred glavnog programa, a ne stavlja se ispred potprograma.

Testiranje jest izvršenje programa ili potprograma s namjerom da se provjeri njegova ispravnost s obzirom na računske rezultate. Pri tome se zadaju takvi podaci, da izvršenje možemo lako kontrolirati.

Pozivanje funkcije vrši se tako da se u aritmetičkom izrazu napiše ime funkcije, a iza toga u zagradi stvarni argumenti.

Stvarni argumenti u pozivu funkcije mogu biti bilo kakvi aritmetički izrazi, te u krajnjoj liniji samo imena varijabli ili konstante.

Podudaranje argumenata mora postojati tako da se stvarni argumenti u pozivu funkcije i formalni argumenti u definiciji funkcije podudaraju po broju, redoslijedu i tipu.

ZAKLJUČAK DRUGOG POGLAVLJA

U drugom poglavlju proširili smo naše znanje, te već možemo lijepo rješavati i složenije zadatke. U prvom redu naučili smo raditi s realnim brojevima, što je naročito potrebno pri rješavanju tehničkih zadataka. Zatim smo naučili kako se brojčani podaci učitavaju i predaju računalu na obradu tek prilikom izvršenja programa. To omogućuje da program bude općenit i da ga koristimo za mnoštvo različitih zadataka istog tipa.

Dalje smo naučili jednu važnu saobraćajnu naredbu za generiranje petlja u programu, tako da se određena rutina ponavlja proizvoljno puta. Usput smo naučili i učitavanje teksta i privremeno zaustavljanje.

Naučili smo također sve o funkcijama, i to bilo kompajlerskim, bilo onima koje mi sami definiramo. To omogućuje da nam program bude kraći i da zauzima manje memorije, jer se po potrebi samo pozivamo na već definiranu rutinu.

I na kraju ovog pogljavlja nalaze se zadaci koji su prilagođeni stečenom znanju. Stečeno znanje još ne omogućuje da te zadatke riješimo na najpovoljniji način koristeći sve mogućnosti FORTRAN-a i samog računala. Ipak se mnoštvo zadataka već može riješiti i s ovim znanjem, a ono što ćemo naučiti u trećem poglavlju, uglavnom će nam olakšati programiranje i, dakako, otvoriti nove putove. I ove zadatke treba riješiti, za njih napisati programe, i to sve izvršiti na elektroničkom računalu, te tako provjeriti ispravnost programa.

ZADACI UZ DRUGO POGLAVLJE

37. Učitaj brojeve p, q pet puta. Izračunaj $p+q$, $p-q$, $p \cdot q$, p/q, p^2, $\sqrt{p}$, $\sin p$, e^p, $\ln p$, $\operatorname{sh} p$.
38. Učitaj a. Ako je $a > 1$, stavi $x = a$, inače $x = 1$. Izračunaj $x^2 + 2x + 3$. Ponovi postupak za razne vrijednosti a, sve dok među podacima ne naiđe prazna kartica (računalo to učita kao $a = 0$.)
39. Učitaj 20 brojeva i izračunaj njihovu sumu.
40. Učitaj cjelobrojnu vrijednost za n. Pomoću DO-petlje izračunaj $n!$ (n faktorijela) radeći sa realnim varijablama.
41. Učitaj cjelobrojnu vrijednost za n. Izračunaj $n!$ i to: 1) ako je $n \leqq 7$, računaj s cijelim brojevima, 2) ako je $7 < n \leqq 33$, računaj s realnim brojevima, 3) ako je $n > 33$, daj obavijest.
42. Učitaj 10 brojeva. Za svaki broj izračunaj sinus hiperbolni i kosinus hiperbolni. Koristi funkcijske naredbe i kompilatorske funkcije.
43. Napiši program s funkcijskim naredbama za tangens, kontangens, arkus sinus, arkus kosinus i u istom programu testiraj te funkcijske naredbe.
44. Učitaj brojeve a, b, c, d. Ako su svi brojevi manji od 25 izračunaj $2a = 3\,b\,/\,c + d$. Inače izračunaj $a+b+c+d$.
45. Učitaj niz od 15 realnih brojeva. Izračunaj sumu njihovih kvadrata.
46. Učitaj niz od 12 realnih brojeva. Otisni najveći broj.
47. Učitaj niz od 12 realnih brojeva. Otisni najmanji broj.
48. Učitaj niz od 12 realnih brojeva. Otisni najveću apsolutnu vrijednost tih brojeva.
49. Učitaj niz od 12 realnih brojeva. Otisni najmanju apsolutnu vrijednost.
50. Učitaj niz od 10 realnih brojeva. Izbroji koliko njih je veće od 5,72.
51. Tabeliraj druge korijene svih cijelih brojeva od 1 do 50.
52. Tabeliraj funkciju sinus za sve kutove s cijelim brojem stupnjeva od 0° do 90°.
53. Učitaj koordinate za pet točaka u prostoru. Odredi njihove udaljenosti od ishodišta.
54. Tiskaj popis učesnika tečaja. Naslov ugradi u program, a datum i imena učitaj prilikom izvođenja programa. Program neka generira i otisne redni broj svakog učesnika.
55. Za niz učitanih realnih brojeva izračunaj i otisni, kao realni broj sa 6 cijelih mjesta i 6 decimala: a) samo dio ispred decimalne točke, b) samo dio iza decimalne točke.
56. Na n kartica dani su podaci za n godina, na svakoj kartici za 12 mjeseci. Učitaj podatke za zadani broj godina i izračunaj sumu po godinama i ukupnu sumu, te prosjek po godinama i ukupni prosjek.
57. Izradi potprogram za učitavanje i izračunavanje srednje vrijednosti proizvoljne količine realnih brojeva, te glavni program za njegovo testiranje.
58. Izradi potprograme za izračunavanje tangensa, odnosno kotangensa zadanog kuta, te glavni program za njihovo testiranje.
59. Izradi potprograme za izračunavanje arkus sinusa, arkus kosinusa i arkus kotangensa zadanog kuta, te glavni program za njihovo testiranje.
60. Izradi program u kojem će biti ugrađena neka uputa operatera, nakon čega treba nastaviti izvršenje programa.

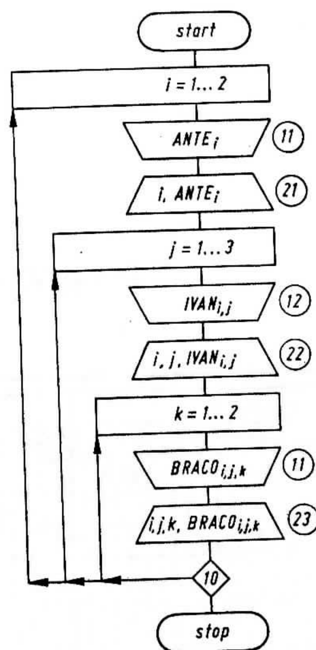
LEKCIJA **31.**

INDEKSIRANE VARIJABLE

```
1    5 6 7  10        20        30        40        50        60
C            SPECIFIKACIJA·POLJA
      DIMENSION ANTE(8),IVAN(5,15),BRACO(10,20,5)
C            JEDNODIMENZIONALNA VARIJABLA
      DO 10 I=1,2
      READ(2,11)ANTE(I)
      WRIT(3,21)I,ANTE(I)
C            DVODIMENZIONALNA VARIJABLA
      DO 10 J=1,3
      READ(2,12)IVAN(I,J)
      WRITE(3,22)I,J,IVAN(I,J)
C            TRODIMENZIONALNA VARIJABLA
      DO 10 K=1,2
      READ(2,11)BRACO(I,J,K)
   10 WRITE(3,23)I,J,K,BRACO(I,J,K)
      CALL EXIT
   11 FORMAT(F10.0)
   12 FORMAT(I10)
   21 FORMAT(' ANTE(' I1 ') =' F10.4)
   22 FORMAT(' IVAN(' I1 ',' I1 ') =' I5)
   23 FORMAT(' BRACO(' I1 ',' I1 ',' I1 ') =' F10.4)
      END
```

Zadatak ove lekcije jest da se učitaju i tiskaju članovi triju polja (indeksirane varijable), pri čemu je jedno polje jednodimenzionalno (varijable s jednim indeksom), drugo polje je dvodimenzionalno (dva indeksa), a treće je trodimenzionalno (tri indeksa). Svrha vježbe je da se uvedu polja, tj. indeksirane varijable.

Shema toka ne sadrži računske operacije, već samo učitavanje i tiskanje. Za učitavanje su korištene tri DO-petlje, pa dakako i ulazni podaci moraju biti poredani na odgovarajući način.




```
     1
    11
   111
   112
    12
   121
   122
    13
   131
   132
     2
    21
   211
   212
    22
   221
   222
    23
   231
   232
```

Ulazni podaci mogli bi biti bilo kakvi. Mi smo međutim uzeli takve ulazne podatke, da nam bude lakše pratiti izlazne rezultate. Zato su ulazni podaci analogni indeksima dotičnih varijabla. Ovako obično postupamo pri testiranju programa, da nam sve bude što preglednije.

Izlazni rezultati nakon izvršenja programa bit će ovog oblika. Izlaz smo programirali tako da budu otisnuta i imena varijabli skupa s indeksima, opet zbog veće preglednosti.

```
ANTE(1) =       1.0000
IVAN(1,1) =     11
BRACO(1,1,1) =  111.0000
BRACO(1,1,2) =  112.0000
IVAN(1,2) =     12
BRACO(1,2,1) =  121.0000
BRACO(1,2,2) =  122.0000
IVAN(1,3) =     13
BRACO(1,3,1) =  131.0000
BRACO(1,3,2) =  132.0000
ANTE(2) =       2.0000
IVAN(2,1) =     21
BRACO(2,1,1) =  211.0000
BRACO(2,1,2) =  212.0000
IVAN(2,2) =     22
BRACO(2,2,1) =  221.0000
BRACO(2,2,2) =  222.0000
IVAN(2,3) =     23
BRACO(2,3,1) =  231.0000
BRACO(2,3,2) =  232.0000
```

Polje je skup većeg broja veličina (elementi polja), koje u polju imaju točno određeni položaj. Položaj pojedinog elementa u polju definiran je indeksom.

Ime polja gradi se na isti način kao što je već rečeno za imena varijabli. Pojedini element polja definiran je imenom polja i indeksom, odnosno indeksima, u zagradi iza imena polja.

Indeksirana varijabla se upotrebljava da se simbolički označi jedan član polja. Sastoji se od imena polja i od indeksa (u zagradi iza imena) koji označava položaj tog člana u polju.

Indeks mora biti cio broj, a vrijednost mu mora biti veća od nule.

Količina indeksa može iznositi do tri: jednodimenzionalno polje ima jedan indeks, dvodimenzionalno polje ima dva indeksa, a trodimenzionalno polje ima tri indeksa.

DIMENSION je organizaciona naredba za specifikaciju polja, pa se mora nalaziti na početku programa. Time se rezervira dio memorije za smještaj polja određene veličine. Ujedno se definira da ime (iza DIMENSION) znači polje (a ne npr. funkciju).

Veličina polja rezervira se u naredbi DIMENSION tako da se iza imena polja napiše potrebna količina indeksa i njihova maksimalna vrijednost, i to pomoću cjelobrojnih konstanata. Rezervacija ne smije biti manja od onoga što se koristi prilikom izvršenja programa.

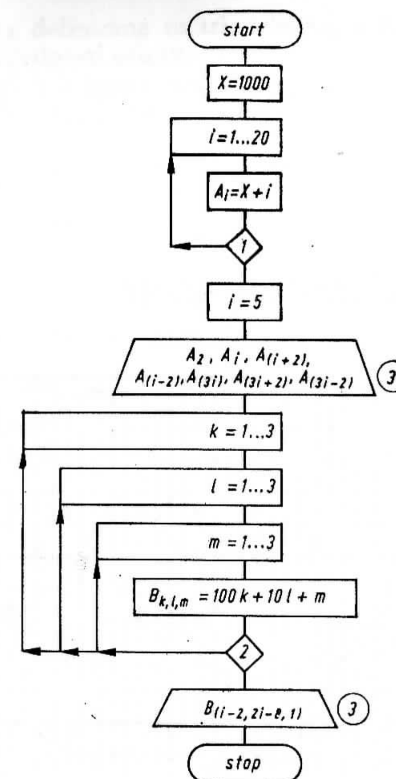
32. LEKCIJA

INDEKSNI IZRAZI

```
C          INDEKSNI IZRAZI
      DIMENSION A(20),B(3,3,3)
C          GENERIRANJE POLJA A
      X=1000.
      DO 1 I=1,20
    1 A(I)=X+I
C          STAMPANJE CLANOVA POLJA A
      I=5
      WRITE(3,3) A(2),A(I),A(I+2),A(I-2),A(3*I),A(3*I+2),A(3*I-2)
C          GENERIRANJE POLJA B
      DO 2 K=1,3
      DO 2 L=1,3
      DO 2 M=1,3
    2 B(K,L,M)=100*K+10*L+M
C          STAMPANJE CLANA POLJA B
      WRITE(3,3) B(I-2,2*I-8,1)
    3 FORMAT(7F10.0)
      CALL EXIT
      END
```

Zadatak ove lekcije jest da se generiraju dva polja i otisnu vrijednosti nekih članova. Svrha je da se upoznamo s izrazima pomoću kojih se mogu definirati indeksi polja. Radi olakšanja rada vrijednosti pojedinih članova polja nisu učitane niti definirane pomoću konstanti, nego su generirane pomoću samog programa.

Shema toka sadrži jednu DO-petlju za generiranje polja A i 3 DO-petlje za generiranje polja B. Tiskaju se vrijednosti samo nekih članova polja. Računskih operacija nema.



Nakon izvršenja programa imat ćemo ovakav izlazni rezulat.

```
   1002.     1005.     1007.     1003.     1015.     1017.     101
    321.
```

Indeksni izraz može se upotrijebiti za definiranje brojčane vrijednosti indeksa unutar programa. Nasuprot tome, u naredbi DIMENSION smije se upotrijebiti samo konstanta.

Oblik indeksnog izraza je strogo određen i ograničen na oblike navedene u ovoj lekciji. Ne smiju se upotrijebiti složeniji oblici niti drugačiji redoslijed varijabli i konstanata.

Vrijednost indeksnog izraza mora uvijek biti cjelobrojna i veća od nule.

Indeksni izrazi za višedimenzionalno polje odvajaju se zarezom.

Generiranje brojčanih vrijednosti za članove nekog polja može se vršiti unutar programa (bez učitavanja!), što je prikladno ne samo pri izradi primjera, već naročito pri testiranju pojedinih rutina.

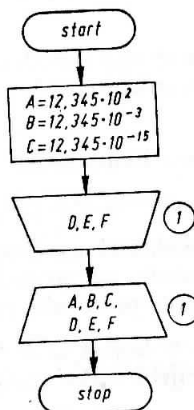
Miješani aritmetički izraz, u kojem se s desne strane nalaze i cjelobrojne i realne veličine, smije se upotrijebiti. Računalo pri tome po potrebi pretvara cjelobrojnu veličinu u realnu.

33. LEKCIJA

EKSPONENCIJALNI OBLIK REALNOG BROJA

```
C         KONSTANTE
      A=12.345E2
      B=12.345E-3
      C=-12.345E-15
C         ULAZNI PODACI
      READ(2,1)D,E,F
    1 FORMAT(3E15.5)
C         IZLAZNI REZULTATI
      WRITE(3,1)A,B,C,D,E,F
      CALL EXIT
      END
```

Zadatak u ovoj lekciji jest da se otisnu 3 brojčane vrijednosti definirane u programu i 3 brojčane vrijednosti učitane tokom izvršenja programa. Sve to treba napisati, učitati i otisnuti u eksponencijalnom obliku. Radi se, dakako, o realnim veličinama. Svrha lekcije jest da se uvede eksponencijalni oblik za realne brojeve. U programu se vidi kako se pišu konstante u eksponencijalnom obliku.



U shemi toka definirane su tri veličine, a zatim se učitavaju tri veličine i konačno se te vrijednosti otisnu.

Ulazni podaci nek imaju ovaj oblik. Ovdje se vidi kako se u eksponencijalnom obliku pišu ulazni podaci.

```
         12.345E2        12.345E-3      -12.345E-15
```

Izlazni rezultati nakon izvršenja programa imaju ovaj oblik. Ovdje se vidi kako se u eksponencijalnom obliku tiskaju izlazni rezultati.

```
   0.12345E 04    0.12345E-01   -0.12345E-13
   0.12345E 04    0.12345E-01    0.12345E-13
```

E-format služi za prikazivanje realnih veličina u eksponencijalnom obliku, tj. pomoću neke veličine kao mantise pomnožene s bazom 10, podignutom na neki eksponent.

Oblik E-formata sastoji se od mantise (pisane kao realna konstanta s točkom), iza toga dolazi veliko slovo E i cjelobrojni eksponent. Mantisa i eksponent mogu biti pozitivni ili negativni. Eksponent ima jednu ili dvije znamenke.

Upotreba E-formata dolazi do izražaja kod veoma malih ili veoma velikih realnih brojeva, gdje bi prikazivanje u F-formatu bilo neprikladno zbog prevelike dužine.

Standardni oblik E-formata ima mantisu koja počinje s nulom i decimalnom točkom, a prva decimala je različita od nule.

Konstante u E-formatu, koje se pišu u programu, ne moraju se pisati u standardnom obliku.

Ulazni podaci u E-formatu ne moraju se pisati u standardnom obliku.

Specifikacija E-formata u naredbi FORMAT sastoji se od slova E, cijelog broja koji definira ukupnu količinu kućica, točke, i cijelog broja koji definira količinu decimalnih mjesta.

Izlazni rezultati u E-formatu tiskaju se uvijek u standardnom obliku. Za eksponent se uvijek utroše tri kućice, i to jedna za predznak (plus se ne otisne, nego ostaje prazno) i dvije za brojčanu vrijednost (po potrebi ispisuje se na prvom mjestu nula).

Broj decimalnih mjesta kod ulaznih podataka ne mora se slagati sa specifikacijom u naredbi FORMAT. U tom slučaju mjerodavno je ono kako je na ulaznim podacima.

Broj decimalnih mjesta kod izlaznih rezultata uvijek se slaže sa specifikacijom u naredbi FORMAT.

Zaokruženje brojčane vrijednosti pri tiskanju izlaznih rezultata se ne vrši. Suvišni broj decimala mantise se odbacuje bez zaokruživanja.

34. LEKCIJA

ČITANJE I TISKANJE POLJA

```
      DIMENSION A(5),B(3,2),C(2,2,2)
C           CITANJE POLJA A
      READ(2,1) A
    1 FORMAT(5F5.0)
C           GENERIRANJE POLJA B
      DO 2 I=1,3
      DO 2 J=1,2
    2 B(I,J)=10*I+J
C           GENERIRANJE POLJA C
      DO 3 I=1,2
      DO 3 J=1,2
      DO 3 K=1,2
    3 C(I,J,K)=100*I+10*J+K
C           STAMPANJE
      WRITE(3,4)(A(I),I=2,5),((B(I,J),J=1,2),I=1,2),C
    4 FORMAT(4F10.1)
      CALL EXIT
      END
```

Zadatak ove lekcije jest da se učita jedno i generiraju dva polja, pa da se to djelomično otisne. Svrha je da se vidi kako se naredbe za čitanje, odnosno tiskanje polja, u potpunosti ili samo djelomično mogu prikladno napisati.

Shema toka sadrži čitanje i generiranje polja, te tiskanje (shema toka je na str. 93).

Ulazni podaci za polje koje se učitava neka ima ovaj oblik.

```
    1    2    3    4    5
```


start

$A_i, i = 1 \ldots 5$ (1)

$i = 1 \ldots 3$

$j = 1 \ldots 2$

$B_{i,j} = 10i + j$

(2)

$i = 1 \ldots 2$

$j = 1 \ldots 2$

$k = 1 \ldots 2$

$C_{ijk} = 100i + 10j + k$

(3)

$A_i, i = 2 \ldots 5$

$B_{ij}, i = 1 \ldots 2, j = 1 \ldots 2$

$C_{ijk}, k = 1 \ldots 2, j = 1 \ldots 2, i = 1 \ldots 2$ (4)

stop

Izlazni rezultati imat će ovaj oblik.

```
       2.0       3.0       4.0       5.0
      11.0      12.0      21.0      22.0
     111.0     211.0     121.0     221.0
     112.0     212.0     122.0     222.0
```

Čitavo polje može se učitati ili otisnuti tako da se u naredbi READ, odnosno WRITE, navede samo ime polja bez ikakvih indeksa. Time se implicitno podrazumijeva čitavo polje, kako je specificirano u naredbi DIMENSION.

Redoslijed članova jednodimenzionalnog polja, ako je čitanje odnosno tiskanje naređeno implicitno za čitavo polje, jest od indeksa 1 do najvećeg indeksa.

Redoslijed članova višedimenzionalnog polja, ako je čitanje, odnosno tiskanje, naređeno implicitno za čitavo polje, jest takav da se najprije mijenja prvi indeks od 1 do najveće vrijednosti, a tek onda se na isti način mijenja drugi indeks, i konačno treći indeks (ukoliko postoji).

Eksplicitna naredba za čitanje, odnosno tiskanje članova polja, omogućuje da se polje otisne samo djelomično i sa proizvoljnim redoslijedom. Uz ime polja treba da stoje varijable kao indeksi, a promjena indeksa se definira analogno kao kod naredbe DO.

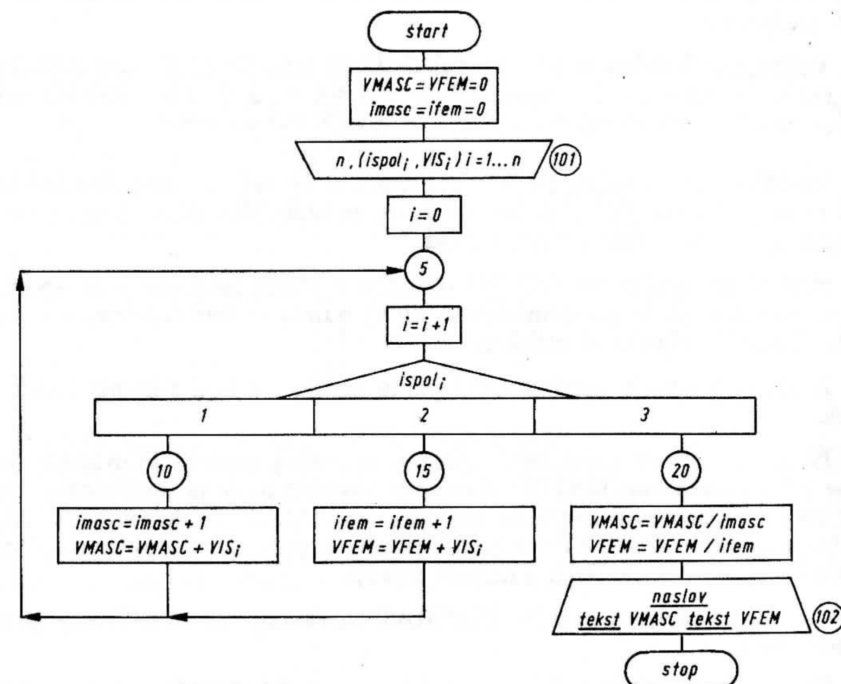
35. LEKCIJA

SKRETNICA, NOVI ZAPIS

```
C          PRORACUN SREDNJE VISINE
      DIMENSION ISPOL(100),VIS(100)
      VMASC=0.
      VFEM=0.
      IMASC=0.
      IFEM=0.
C          UCITAVANJE, SKRETNICA
      READ(2,101)N,(ISPOL(I),VIS(I),I=1,N)
      I=0
    5 I=I+1
      J=ISPOL(I)
      GO TO(10,15,20),J
C          STAZA ZA MUSKE
   10 IMASC=IMASC+1
      VMASC=VMASC+VIS(I)
      GO TO 5
C          STAZA ZA ZENSKE
   15 IFEM=IFEM+1
      VFEM=VFEM+VIS(I)
      GO TO 5
C          STAZA ZAVRSETAKA
   20 VMASC=VMASC/IMASC
      VFEM=VFEM/IFEM
      WRITE(3,102)VMASC,VFEM
      CALL EXIT
  101 FORMAT(I5/(I1,F9.0))
  102 FORMAT(///' SREDNJA VISINA IZNOSI'
     */' ZA MUSKE' E15.7 /' ZA ZENSKE' E14.7)
      END
```


Zadatak u ovoj lekciji jest da se učita kup kartica, svaka sa oznakom spola (1 = muško, 2 = žensko) i visinom. Na početku kupa je kartica s količinom kartica i s tekstom, koji ne treba učitati. Od svih visina muških treba izračunati srednju vrijednost i analogno za visine ženskih.

Shema toka ima tri paralelne staze: jednu za brojenje koliko je muških visina učitano i za sumiranje svih muških visina, drugu analognu za ženske, i treću za proračun srednjih visina i tiskanje. Za izbor staze korištena je skretnica, a njom upravlja učitana oznaka spola. Za završetak je umjesto oznake spola predviđeno učitavanje brojke 3, kako bi se izvršenje programa uputilo na treću stazu.



Prva kartica ima jedan ulazni podatak i tekst za objašnjenje koji neće biti učitan. Dalje slijede kartice s oznakom spola i visinom.

```
    6 KARTICA S PODACIMA
1      183
1      175
2      159
1      178
2      162
3
```

Izlazni rezultati imat će ovaj oblik:

```
SREDNJA VISINA IZNOSI
ZA MUSKE  0.1786666E 03
ZA ZENSKE 0.1605000E 03
```

Izačunata GO TO naredba je saobraćajna naredba koja služi kao skretnica, jer omogućuje skok na jednu od više naredaba po izboru, već prema izračunatoj vrijednosti indeksa. Iza GO TO dolazi u zgradi niz brojeva naredaba na koje može biti izveden skok. Na kraju se nalazi indeks dan cjelobrojnom varijablom.

Vrijednost indeksa u izračunatoj GO TO naredbi može biti definirana u programu, izračunata ili učitana. Može iznositi 1, 2, 3 itd. i najviše onoliko, koliko ima brojeva naredaba u zagradi te GO TO naredbe.

SKOK nakon izračunate GO TO naredbe se vrši: na naredbu čiji je broj prvi u zagradi, ako je ideks jednak 1, na naredbu čiji je broj drugi po redu u zagradi, ako je indeks jednako 2 itd.

Simbol za izračunatu GO TO naredbu u shemi toka može se upotrijebiti kao u ovoj lekciji. U gornjem dijelu (krov) nalazi se ime indeksa, a u donjem dijelu (kućice) vrijednosti indeksa.

Zapis kod čitanja kartica znači jednu karticu, a kod tiskanja znači jedan redak.

Naredba za nov zapis implicirana je u svakoj naredbi FORMAT (uz naredbe READ, odnosno WRITE). Možemo zamisliti da je ta implicirana naredba smještena u »otvorenoj zagradi« iza riječi FORMAT. Svaki put kad dođe do izvršenja naredbe READ ili WRITE računalo prelazi na nov zapis (čita slijedeću karticu, odnosno tiska u novom retku).

/ je specifikacija u naredbi FORMAT za eksplicitno naređivanje prijelaza na nov zapis.

Prekratak FORMAT (više varijabla u naredbi READ, odnosno WRITE nego što je specificirano u naredbi FORMAT) dovodi do ponavljanja svih specifikacija počev od lijeve otvorene zagrade. Tim je ujedno (implicitno) dana naredba za prijelaz na nov zapis.

Zagrade iza / u naredbi FORMAT dovode do ponavljanja samo onih specifikacija koje se nalaze u tim zagradama, opet svaki put s prijelazom na nov zapis.

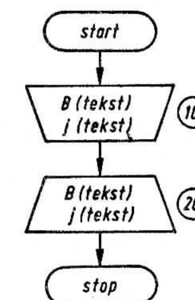
LEKCIJA **36.**

TEKST KAO VARIJABLA

```
      DIMENSION B(6), J(3)
C     UCITAVANJE
      READ(2,10) B,J
   10 FORMAT(6A4/3A2)
C     STAMPANJE
      WRITE(3,20) B,J
   20 FORMAT(1X,6A4,1X,3A2)
      CALL EXIT
      END
```

Zadatak ove lekcije jest da se učita i otisne neki tekst. Međutim, ovdje se traži učitavanje na jedan, a otisne na drugi način. To je moguće učiniti pomoću naredaba koje tretiraju tekst kao varijablu. Takvo tretiranje teksta stvara i daljnje mogućnosti za čitanje i tiskanje.

U shemi toka se nalazi samo čitanje i tiskanje, ali je označeno da su to varijable koje ne sadrže brojčane vrijednosti, nego tekst.



Ulazni podaci sadržavaju tekst u dva retka (dvije kartice).

```
ELEKTROTEHNICKI FAKULTET
ZAGREB
```

Izlazni rezultati sadržavaju tekst u jednom retku.

```
ELEKTROTEHNICKI FAKULTET ZAGREB
```

A-format kao specifikacija u naredbi FORMAT omogućuje čitanje i tiskanje teksta koji se pridaje varijabli (analogno kao što se pridaje brojčana vrijednost).

Realnoj varijabli mogu se A-formatom pridati 4 alfanumerička znaka (slova, brojke, specijalni znakovi). U tom slučaju u specifikaciji iza slova A stoji brojka 4.

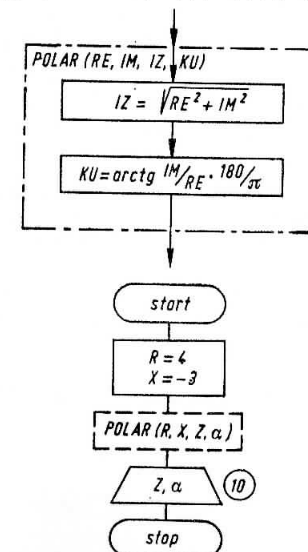
Cjelobrojnoj varijabli mogu se A-formatom pridati 2 alfanumerička znaka. Tada u specifikaciji u naredbi FORMAT iza slova A stoji brojka 2.

Duži tekst, sa više od 2 znaka, može se pridati nizu varijabli (obično polju s odgovarajućim brojem članova). U tom slučaju ispred slova A mora stajati brojka jednaka količini varijabla (odnosno količini članova polja).

LEKCIJA **37.**

OPĆENITI POTPROGRAM

```
C         DEFINICIJA POTPROGRAMA
C     POLARNI OBLIK KOMPLEKSNOG BROJA
      SUBROUTINE POLAR (AREAL,AIMAG,AIZNS,AKUT)
C     -----------------
      AIZNS=SQRT(AREAL**2+AIMAG**2)
      AKUT=ATAN(AIMAG/AREAL)*180./3.14159
      RETURN
      END
C         GLAVNI PROGRAM
      R=4.
      X=-3.
      CALL POLAR (R,X,Z,ALFA)
      WRITE (3,10) Z,ALFA
   10 FORMAT(' Z =',F5.2,' OMA SA',F7.2,' STUPNJEVA')
      CALL EXIT
      END
```



Zadatak je ove lekcije da se u okviru potprograma izračuna kompleksna veličina u polarnom obliku (iznos i kut), ako su poznate realna i imaginarna komponenta. Osim toga treba izraditi glavni program za testiranje potprograma. Svrha lekcije jest da se uvede općenit potprogram, u kojem nema ograničenja kao kod funkcije, odnosno funkcijske naredbe.

Shema toka ovdje sadrži posebno shemu toka za definiciju potprograma (s formalnim argumentima) i posebno shemu toka za glavni program, u kojem se nalazi poziv potprograma (sa stvarnim argumentima).

Izlazni rezultati ovog su oblika.

```
Z = 5.00 OMA SA -36.86 STUPNJEVA
```

Općeniti potprogram je takav potprogram koji može imati proizvoljni broj naredaba, a isto tako i proizvoljni broj rezultata.

SUBROUTINE je organizaciona naredba koja stoji na početku definicije općenitog potprograma zajedno s imenom potprograma i formalnim argumentima.

Ime općenitog programa gradi se na isti način kao imena varijabli, ali prvo slovo nema nikakvog značenja (u pogledu definiranja realne ili cjelobrojne veličine), jer uz ime nije vezana nikakva veličina. Ime je jedina veza između definicije i poziva potprograma, i zato mora biti isto u definiciji i u pozivu.

Formalni argumenti nalaze se u definiciji potprograma u zagradi iza imena, međusobno odvojeni zarezima. Argumenti mogu biti ulazni (preko njih se unose brojčani podaci u potprogram) ili izlazni (preko njih se mogu koristiti rezultati izračunati u potprogramu). Argumenti mogu biti istodobno ulazni i izlazni.

Imena formalnih argumenata u definiciji nemaju nikakve veze s imenima izvan potprograma, pa pri njihovu izboru postoji potpuna sloboda.

Definicija potprograma sadrži potrebne naredbe za izračunavanje rezultata. Niti brojevi naredaba niti imena varijabli u definiciji nemaju nikakve veze s imenima izvan potprograma, pa pri njihovu izboru postoji potpuna sloboda.

U definiciji se pojavljuju svi argumenti, i to ulazni u aritmetičkim izrazima (s desne strane znaka jednakosti), a izlazni s lijeve strane znaka jednakosti u aritmetičkim naredbama. Tu se pojavljuje bar jednom naredba RETURN, i na kraju definicije naredba END.

CALL je izvršna naredba za poziv općenitog potprograma. Iza riječi CALL dolazi ime potprograma i u zagradi iza toga stvarni argumenti. Može se pozivati samo onaj potprogram koji se već nalazi spremljen na disku.

Stvarni argumenti u naredbi CALL moraju se sa formalnim argumentima u naredbi SUBROUTINE slagati po broju, redoslijedu i tipu (realni, odnosno cjelobrojni).

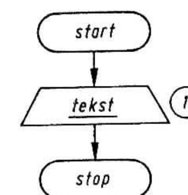
Stvarni ulazni argumenti moraju prije poziva potprograma biti poznati (učitani ili izračunati).

LEKCIJA **38.**

POMAK PAPIRA PRI TISKANJU

```
C         STAMPANJE TEKSTA
      WRITE(3,1)
    1 FORMAT('1NOVA STRANICA'
     *      /'0NOVI REDAK S PROREDOM'
     *      /' NOVI REDAK'
     *      /'+             ISTI REDAK')
      CALL EXIT
      END
```

Zadatak ove lekcije jest da se tiska neki tekst, ali uz različit pomak papira. Pri tome se može postići prijelaz na novu stranicu, nov redak sa proredom, nov redak bez proreda ili tiskanje u istom retku.



Shema toka sadrži samo općeniti simbol za tiskanje teksta. Tekst koji će se tiskati i njegov raspored treba dati posebno. To se po potrebi može nacrtati na posebnom rasteru.

Izlazni rezultat ima ovaj oblik (nakon 2 prazna retka na novoj stranici):

```
NOVA STRANICA

NOVI REDAK S PROREDOM
NOVI REDAK  ISTI REDAK
```

Pomak papira po visini postiže se kod linijske tiskaljke pomoću naredbe za tiskanje prvog znaka u retku.

Prvi znak u retku se ne otisne, već služi samo za upravljanje pomakom papira, bez obzira kako je dana naredba za njegovo tiskanje.

1 kao prvi znak u retku služi kao eksplicitna naredba za prijelaz na novu stranicu prije tiskanja tog retka. (U stvari je to skok do rupice koju izbušimo na 1. kanalu trake vodilice, a taj kanal se redovito koristi za prijelaz na novu stranicu).

0 kao prvi znak u retku služi kao eksplicitna naredba za prijelaz u novi redak sa proredom.

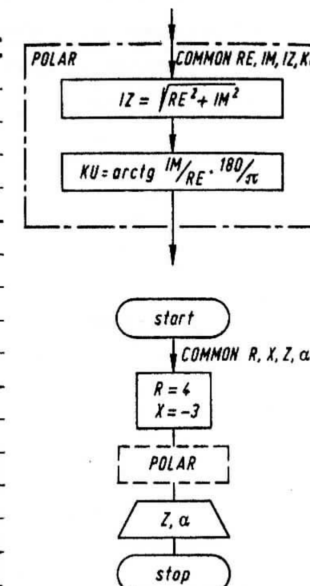
Bjelina kao prvi znak u retku ostavlja na snazi implicitnu naredbu za prijelaz na nov zapis, a to je u ovom slučaju novi redak bez proreda.

+ kao prvi znak u retku služi kao eksplicitna naredba za tiskanje u istom retku. U stvari, time se poništava implicitna naredba za prijelaz za novi zapis.

Automatski prijelaz za novu stranicu vrši računalo samo od sebe kad stigne do kraja stranice. (U stvari to se događa kad stignemo do rupice na 12. kanalu trake vodilice, koju rupicu izbušimo po želji.)

POISTOVEĆIVANJE VARIJABLA

```
C        POTPROGRAM
      SUBROUTINE POLAR
      COMMON AREAL,AIMAG,AIZNS,AKUT
      AIZNS=SQRT(AREAL**2+AIMAG**2)
      AKUT=ATAN(AIMAG/AREAL)*180./3.14159
      RETURN
      END
C        GLAVNI PROGRAM
      COMMON R,X,Z,ALFA
      R=4.
      X=-3.
      CALL POLAR
      WRITE (3,10) Z,ALFA
10    FORMAT(' Z =',F5.2,' SA',F7.2)
      CALL EXIT
      END
```



Zadatak u ovoj lekciji je isti kao i lekciji 37. Razlika između ove lekcije i lekcije 37 je samo u tome što ćemo ovdje zadatak riješiti na drugačiji način. Svrha nam je da uvedemo mogućnost poistovećivanja varijabli, koje na raznim mjestima nose različita imena.

Shema toka je ista kao u lekciji 37, samo što je ovdje označeno da će se upotrijebiti poistovećivanje varijabla.

Izlazni rezultat ima ovaj oblik.

```
Z = 5.00 SA -36.86
```

COMMON je organizaciona naredba za poistovećivanje varijabla između različitih potprograma, odnosno glavnih programa. Iza riječi COMMON dolaze imena varijabli koje se poistovećuju.

Varijable iza COMMON moraju se slagati po redoslijedu i tipu (realna ili cjelobrojna). Poistovećuju se sve prve varijable međusobno, sve druge međusobno itd.

Ime polja iza COMMON piše se bez indeksa, ako je polje definirano u naredbi DIMENSION. Smiju se poistovetiti samo polja istih dimenzija i veličine.

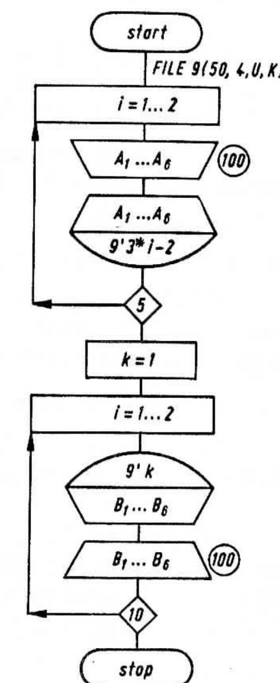
Argumenti (formalni i stvarni) koji su navedeni u naredbi COMMON, ne smiju se ponoviti u definiciji potprograma niti u pozivu potprograma.

Smještaj naredbe COMMON jest na početku programa (odnosno na početku potprograma iza naredbe SUBROUTINE), ali iza naredbe DIMENSION (ukoliko ta postoji).

LEKCIJA **40.**

SPREMANJE BROJČANIH VRIJEDNOSTI NA DISK

```
      DIMENSION A(6), B(6)
      DEFINE FILE 9(50,4,U,K)
C          CITANJE I SPREMANJE NA DISK
      DO 5 I=1,2
      READ (2,100) A
    5 WRITE (9'3*I-2) A
               UZIMANJE S DISKA I STAMPANJE
      K=1
      DO 10 I=1,2
      READ (9'K) B
   10 WRITE (3,100) B
      CALL EXIT
  100 FORMAT(6F5.0)
      END
```



Zadatak u ovoj lekciji jest da se učitaju podaci s kartica i spreme na disk. Zatim da se uzmu s diska i otiskuju. Pri tome se postavlja ograničenje da se ne učitaju svi podaci pa onda spremaju na disk, nego da se to radi postepeno. Isto kod uzimanja s diska i tiskanja. Na taj način može se mnogo podataka spremiti na disk, a da se u brzoj memoriji zauzme samo mali dio memorije.

Shema toka sadrži dvije DO-petlje. U prvoj se ulazni podaci kao članovi polja A postepeno spremaju na disk. U drugoj petlji se postepeno te brojčane vrijednosti uzimlju s diska, pridaju polju B (mogli smo upotrijebiti i isto polje) i otisnu. Polja A i B imaju po 6 članova. Pretinac na disku podijeljen je u pregrade koje imaju po 4 riječi, tako da u svaku pregradu stanu po 2 realne veličine (svaka zauzima prostor od 2 riječi).

Ulazni podaci neka su zadani ovako:

```
    1    2    3    4    5    6
    7    8    9   10   11   12
```

Izlazni rezultati bit će ovog oblika:

```
 1.  2.  3.  4.  5.  6.
 7.  8.  9. 10. 11. 12.
```

Disk predstavlja vanjsku memoriju sa 512000 riječi. Svaka brojčana veličina, realna i cjelobrojna, zauzima po 2 riječi. Pri spremanju brojčanih vrijednosti na disk moraju se najprije na disku organizirati pretinci (FILE).

Pretinac na disku nosi svoju numeraciju, a podijeljen je na pregrade, svaka s određenim brojem riječi. Za identifikaciju pregrade može se (ali ne mora) koristiti kazalo tog pretinca.

DEFINE FILE je organizaciona naredba za organizaciju pretinca na disku. Tu se definira numeracija pretinca, količina pregrada, veličina pregrada i ime kazala. Može se koristiti i za organizaciju većeg broja pretinaca.

Numeracija pretinca dolazi iza riječi FILE. To je pozitivna cjelobrojna konstanta (između 1 i 32767).

Količina pregrada dolazi kao prva u zagradi iza numeracije pretinca. To je cjelobrojna pozitivna konstanta.

Veličina pregrada dolazi kao druga u zagradi. To je cjelobrojna pozitivna konstanta, koja može najviše iznositi 320. Veličina pregrade znači broj riječi u pregradi, i za svaku brojčanu veličinu treba predvidjeti 2 riječi.

U dolazi na treće mjesto u zagradi i treba ga doslovno napisati.

Kazalo dolazi na četvrto mjesto u zagradi. To je cjelobrojna varijabla bez indeksa. Isto ime varijable može se koristiti i za veći broj pretinaca.

WRITE (.......' ....) je izvršna naredba za spremanje brojčanih vrijednosti na disk. Na prvom mjestu u zagradi nalazi se numeracija pretinca (cjelobrojna konstanta ili cjelobrojna varijabla). Na drugom mjestu u zagradi nalazi se numeracija pregrade u koju se počinje spremati (cjelobrojni aritmetički izraz).

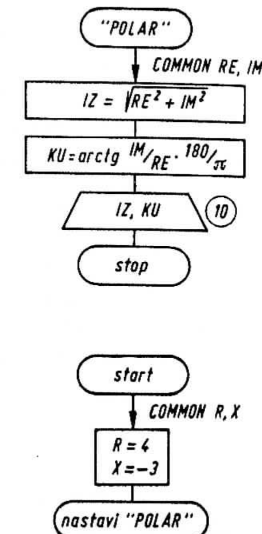
READ (.......' ....) je izvršna naredba za uzimanje brojčanih vrijednosti diska, analogno naredbi WRITE (.......' ....).

Upotreba kazala olakšava definiranje početne pregrade. Prije naredbe za spremanje ili uzimanje s diska kazalu treba pridati brojčanu vrijednost, a zatim ime kazala upisujemo iza apostrofa u odnosnu naredbu. Pri izvršenju spremanja, odnosno uzimanja podataka s diska, vrijednost kazala se automatski postavlja na numeraciju prve slijedeće nekorištene pregrade.

LEKCIJA **41.**

POZIV VEZANOG PROGRAMA

```
C         VEZANI PROGRAM POLAR
      COMMON AREAL,AIMAG
      AIZNS=SQRT(AREAL**2+AIMAG**2)
      AKUT=ATAN(AIMAG/AREAL)*180./3.14159
      WRITE (3,10) AIZNS,AKUT
10    FORMAT(' Z =' F5.2 ' SA' F7.2)
      CALL EXIT
      END
C         GLAVNI PROGRAM
      COMMON R,X
      R=4.
      X=-3.
      CALL LINK(POLAR)
      END
```



Zadatak u ovoj lekciji je isti kao u lekciji 37, ali ga treba riješiti u dva odvojena programa, koji su međusobno vezani. Svrha je lekcije da upoznamo ovaj način programiranja, pri čemu se u glavnoj memoriji nalazi samo onaj program koji se upravo izvršava. Izvršenje počinje s glavnim programom, a nastavlja se s vezanim programom, koji već prije mora biti spremljen na disk.

Shema toka sadrži dva odvojena programa. Prvi, koji treba spremiti na disk pod imenom »POLAR« i s kojim se izvršenje završava. I drugi, s kojim se izvršenje započinje, a koji poziva vezani program »POLAR« kao nastavak. Potrebne varijable između oba programa su poistovećene.

Izlazni rezultat ovog je oblika.

```
Z = 5.00 SA -36.86
```

CALL LINK je izvršna saobraćajna naredba za pozivanje vezanog programa na izvršenje. Ime vezanog programa nalazi se u zagradi iza riječi LINK.

Smještaj CALL LINK naredbe je na kraju izvršnih naredbi u glavnom programu, tamo gdje bi se inače nalazila naredba STOP, odnosno CALL EXIT.

Vezani program je onaj koji pozivamo pomoću naredbe CALL LINK. On se već mora nalaziti spremljen na disku pod svojim imenom.

Ime vezanog programa gradi se kao i ime varijabli, ali prvo slovo nema posebno značenje (isto kao ime općenitog potprograma).

Poistovećenje varijabla potrebno je između glavnog programa i vezanog programa za one varijable koje se pojavljuju u oba programa.

Izbor imena varijabla i brojeva naredaba u oba programa je potpuno slobodan.

42. LEKCIJA

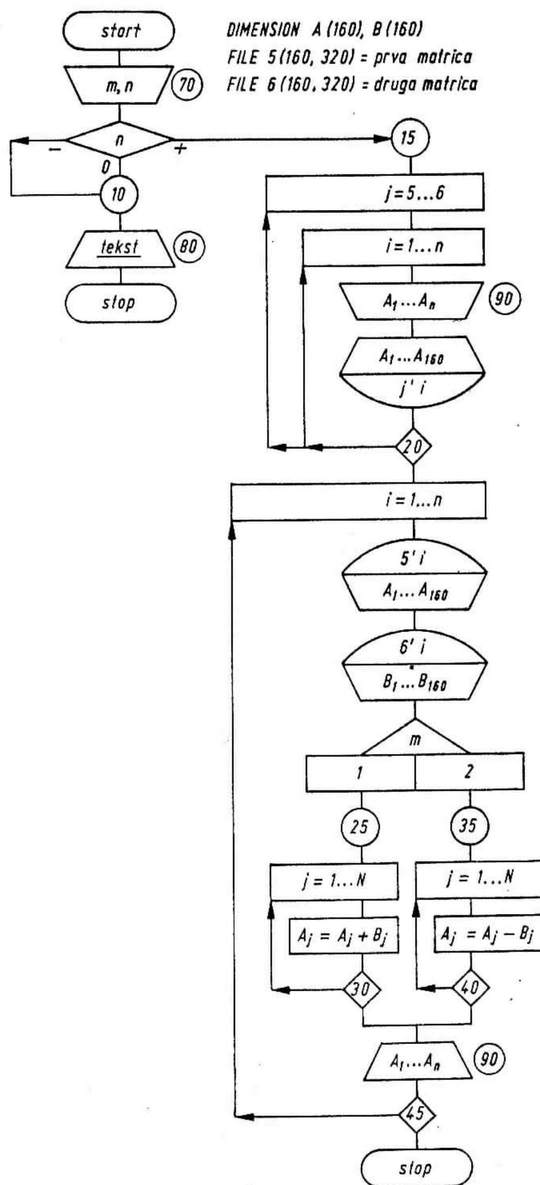
MONITOR I TESTIRANJE

Zadatak ove lekcije jest učitavanje dviju velikih kvadratnih matrica, te njihovo zbrajanje ili odbijanje, i konačno tiskanje rezultata. Zbog veličine matrica treba koristiti disk za spremanje brojčanih veličina. Ako učitani red matrice nije u redu (mora biti veći od nule), o tome treba dati obavijest. Za vrstu računanja učitati karakteristiku, i to: 1 za zbrajanje, a 2 za odbijanje. U program treba uključiti ostavljanje aritmetičkog i saobraćajnog traga, a pri izvršenju programa to treba koristiti.


```
// JOB T              STEFANINI ETF
// FOR
*IOCS(CARD,1132PRINTER,DISK)
*LIST SOURCE PROGRAM
*ARITHMETIC TRACE
*TRANSFER TRACE
C               VELIKE MATRICE
      DIMENSION A(160),B(160)
      DEFINE FILE 5(160,320,U,K),6(160,320,U,K)
      READ(2,70)M,N
      IF(N)10,10,15
C               RED MATRICE POGRESAN
10    WRITE(3,80)
      CALL EXIT
C               RED MATRICE ISPRAVAN, CITANJE,SPREMANJE
15    DO 20 J=5,6
      DO 20 I=1,N
      READ(2,90)(A(L),L=1,N)
20    WRITE(J'I) A
C               UZIMANJE S DISKA
      DO 45 I=1,N
      READ(5'I) A
      READ(6'I) B
      GO TO(25,35),M
C               ZBRAJANJE MATRICA
25    DO 30 J=1,N
30    A(J)=A(J)+B(J)
      GO TO 45
C               ODBIJANJE MATRICA
35    DO 40 J=1,N
40    A(J)=A(J)-B(J)
45    WRITE(3,90)(A(L),L=1,N)
      CALL EXIT
70    FORMAT(2I5)
80    FORMAT(' RED MATRICE POGRESAN')
90    FORMAT(3F10.1)
      END
// *            UPUTA - UKLJUCI SKLOPKU 15
// PAUS
// XEQ
```

Shema toka sadrži monitorske naredbe uz dio za programiranje u FORTRAN-u. Uz shemu je bilješka za naredbu DIMENSION i za definiranje pretinaca. Program sadrži uvjet za ispitivanje učitanog reda matrice. Ako je ispravan, učitava se najprije prva matrica, i to redak po redak, i svaki redak se odmah spremi na disk. Zatim se analogno učitava i sprema druga matrica. U

MONITOR: 1) Izvedi posao
2) Učitaj FORTRAN – kompajler
3) Uključi aritmetički trag
4) Uključi saobraćajni trag



MONITOR: 5) Uputa – uključi sklopku 15
6) Izvrši program

drugom dijelu sheme toka se sa diska uzima redak po redak jedne i druge matrice. Već prema tome treba li zbrajati ili odbijati, u koju svrhu je upotrijebljena skretnica, odgovarajući članovi se zbrajaju, odnosno odbijaju, i to se odmah otisne kao redak matrice s rezultatom.

Ulazni podaci sadrže najprije karakteristiku (i to 1 za zbrajanje), zatim red matrice (3), te tri retka prve matrice i tri retka druge matrice.

```
    1    3
        1Ø1.      1Ø2.      1Ø3.
        1Ø4.      1Ø5.      1Ø6.
        1Ø7.      1Ø8.      1Ø9.
          1.        2.        3.
          4.        5.        6.
          7.        8.        9.
```

Izlazni rezultati ovog su oblika. Tu se rezultati tiskaju ovako:

a) rezultati otisnuti zbog WRITE u programu — bez zvijezdica,

b) rezultati aritmetičkih naredaba otisnuti zbog naredbe ARITHMETIC TRACE — s jednom zvjezdicom,

c) rezultat izraza u naredbi IF otisnut zbog naredbe TRANSFER TRACE, — sa dvije zvjezdice,

d) vrijednost indeksa GO TO naredbe otisnut zbog naredbe TRANSFER TRACE — sa tri zvjezdice.

```
**       3
***       1
*  0.1020000E 03
*  0.1040000E 03
*  0.1060000E 03
      102.0         104.0         106.0
***       1
*  0.1080000E 03
*  0.1100000E 03
*  0.1120000E 03
      108.0         110.0         112.0
***       1
*  0.1140000E 03
*  0.1160000E 03
*  0.1180000E 03
      114.0         116.0         118.0
```

* IOSC (......., DISK, ...) je FORTRAN-ska upravljačka naredba koja u brzu memoriju smještava sve monitorske rutine potrebne za spremanje brojčanih veličina na disk.

* ARITHMETIC TRACE je FORTRAN-ska upravljačka naredba, da se prilikom izvršenja programa ostavi aritmetički trag.

* TRANSFER TRACE je FORTRAN-ska upravljačka naredba, da se prilikom izvršenja programa ostavi saobraćajni trag.

Sklopka 15, koja se nalazi na konzoli računala, mora biti uključena da bi naredbe * ARITHMETIC TRACE i * TRANSFER TRACE bile izvršene. Ako je isključena, trag se ne ostavlja, iako odnosne naredbe u učitanom programu postoje. Sklopkom 15 barata operater.

Aritmetički trag jest rezultat svake aritmetičke naredbe tokom izvršenja programa. Tiska se s jednom zvjezdicom na početku.

Saobraćajni trag jest rezultat izvršenja bilo naredbe IF (otisne se vrijednost izraza u toj naredbi i dvije zvjezdice na početku), bilo izračunate GO TO naredbe (otisne se vrijednost indeksa u toj naredbi i tri zvjezdice na početku).

// * je monitorska naredba za komentar, koji se može pisati u sve kućice iza zvjezdica.

// **PAUS** je monitorska naredba za privremeno zaustavljanje računala tokom čitanja monitorskih naredbi. Računalo će nastaviti rad kad operater pritisne na tipku PROGRAM START.

ZAKLJUČAK TREĆEG POGLAVLJA

U dosadašnja tri poglavlja obuhvatili smo sve što je bitno u FORTRAN-u za IBM 1130. Ostale su još samo neke finese, koje se rjeđe upotrebljavaju, i bez kojih se također može programirati. O MONITOR-u IBM 1130 naučili smo najvažnije naredbe, tako da možemo raditi s programima i potprogramima pisanima u FORTRAN-u. MONITOR sadrži još mnoge naredbe, a kako je njegova svrha da rastereti operatera, to se najviše naredaba odnosi na vezu operater — računalo. Nije bila niti svrha da to ovdje obuhvatimo.

I na kraju ovog poglavlja nalaze se zadaci koje treba riješiti, napisati za njih programe i testirati ih na elektroničkom računalu. No svrha ovih zadataka je samo da potaknu na rad, jer se sa stečenim znanjem mogu lijepo rješavati praktički svi zadaci s kojima se možemo sresti. Zbog toga može svatko za sebe odabrati bilo kakve zadatke, te ih rješavati. Ipak je preporučljivo da se u početku rješavaju jednostavni zadaci, i da se tek postepeno prijeđe na složene zadatke.

U dosadašnjim lekcijama ujedno smo naučili da zadatke treba najprije formulirati, naći matematičku podlogu za rješavanje, te to sve prikazati u shemi toka. Zatim to sve treba napisati u FORTRAN-u, pa testirati na računalu i tražiti pogreške, eventualno uz korištenje ostavljanja traga. Konačno se ispravni program može korisiti, i to proizvoljno puta. Shema toka za čitav ovaj postupak dana je u prilogu 3.

Konačno je zbog informacije u prilogu 4 dana funkcionalna shema jednostavnog računskog centra s elektroničkim računalom IBM 1130, kakav centar postoji na Elektrotehničkom fakultetu u Zagrebu.

ZADACI UZ TREĆE POGLAVLJE

61. Izradi potprogram za rješavanje kvadratne jednadžbe $ax^2 + bx + c = 0$ s konačnim koeficijentima.
62. Izradi program za testiranje potprograma iz zadataka 61.
63. Isto kao zadatak 61, ali s time da koeficijenti mogu imati vrijednost nule.
64. Isto kao 62, ali za potprogram iz zadataka 63.
65. Izradi potprogram za izračunavanje srednje vrijednosti od n realnih brojeva, i glavni program za testiranje.
66. Izradi potprogram za rješavanje dviju linearnih jednadžbi, i program za testiranje.
67. Izradi potprogram za množenje dvaju vektora (matrica s jednim retkom), i testiraj ga.
68. Izradi potprogram za množenje dviju kvadratnih matrica, i testiraj ga.
69. Izradi potprogram za množenje dviju pravokutnih matrica, i testiraj ga.
71. Izradi potprogram za dijeljenje dvaju kompleksnih brojeva, i testiraj gà.
72. Izradi potprogram za pretvaranje kompleksnog broja (realni i imaginarni član) u polarni oblik, uvaživši da pojedine komponente mogu biti jednake nuli, i da rezultat mora definirati i kvadrant. Testiraj ga.
73. Izradi potprogram za pretvaranje kompleksnog broja iz polarnog oblika u oblik s realnim i imaginarnim članom. Testiraj ga.
74. Otisni potvrde za osobe koje su se prijavile za tečaj. Razlikuj u tekstu mušku osobu, žensku osobu i poduzeće (srednji rod).
75. Učitaj imena triju gradova i tiskaj ih u raznim kombinacijama, svaka kombinacija u novom retku.
76. Izradi potprogram za dijeljenje dvaju cijelih brojeva (po srednjoškolskom pravilu) točno na proizvoljan broj decimala i rezultat otisni u jednom retku, i to sa decimalnim zarezom, a ne točkom.
77. U jednom glavnom programu učitaj 10 realnih brojeva s jedne kartice, a u drugom (vezanom) programu otisni te brojeve na novoj stranici, jedan ispod drugoga s proredom.
78. Izradi potprogram za množenje velikih matrica. Matrice su kvadratne i simetrične, a smještene su na disk. Rezultat smjesti na disk.
79. Izradi program za testiranje potprograma iz zadatka 78.
80. Izračunaj produkt kvadratne matrice i vektora. Otisni naslov, ulazne podatke i rezultat.
81. Učitaj tabelarno neku funkciju, npr. sinus za $x = 0, 10, 20, \dots 90$ stupnjeva. Otisni vrijednost funkcije uz linearnu interpolaciju za neku vrijednost argumenta, npr. za $x = 43{,}5$ stupnjeva.
82. Izradi potprogram za zadatak kao u lekciji 37, ali samo sa dva argumenta (oba ulazno-izlazna) i bez naredbe COMMON. Testiraj ga.
83. Učitaj imena nekoliko gradova (najviše 10) spremi ih na disk, uzmi s diska i otisni.
84. Učitaj dvodimenzionalno polje, preračunavanjem indeksa pretvori ga u jednodimenzionalno i spremi na disk. Učitaj ga s diska, pretvori natrag u dvodimenzionalno i otisni.

LITERATURA

1. IBM 1130/1800 Basic FORTRAN IV Language (Form C 26—3715—0), IBM Co. 1967.
2. IBM 1130 Disk Monitor System, Reference Manual (Form C 26 — 3750 — 1), IBM Co 1967.
3. Stefanini B., FORTRAN Podsjetnik, Elektrotehnički fakultet u Zagrebu, 1969.

FORTRAN

S R C E

SVEUČILIŠTE U ZAGREBU
RAČUNSKI CENTAR

Programer | Problem | Str.
Oznaka | Datum | Od

1 5 6 7 10 20 30 40 50 60 70 72 73 80

FORTRAN — Popis pogrešaka

C 01 Numerički znak u broju naredbe.
C 02 Više nego 5 kontinuiranih kartica, ili kontinuirana kartica izvan niza.
C 03 Sintaktička pogreška u naredbi CALL LINK ili CALL EXIT.
C 04 Neodređena, krivo napisana ili neispravno formirana naredba.
C 05 Naredba izvan redoslijeda.
C 06 Naredba koja slijedi iza naredbe za skok ili za naredbe STOP nema broja.
C 07 Ime duže od 5 znakova, ili ne počinje alfabetskim znakom.
C 08 Indeks unutar naredbe o dimenziji (DIMENSION, COMMON ili TYPE) je neispravan ili ga nema.
C 09 Dupliciran broj naredbe.
C 10 Sintaktička pogreška u naredbi COMMON.
C 11 Duplicirano ime u naredbi COMMON.
C 12 Sintaktička pogreška u naredbi FUNCTION ili SUBROUTINE.
C 13 Formalni argument se pojavljuje u naredbi COMMON.
C 14 Ime se pojavljuje 2 puta kao argument u naredbi SUBROUTINE ili FUNCTION.
C 15 Upravljačka naredba IOCS u potprogramu.
C 16 Sintaktička pogreška u naredbi DIMENSION.
C 17 Ime potprograma u naredbi DIMENSION.
C 18 Ime dimenzionirano više nego jedanput, ili nije dimenzionirano kad se prvi put pojavilo.
C 19 Sintaktička pogreška u naredbi REAL, INTEGER ili EXTERNAL.
C 20 Ime potprograma u naredbi REAL ili INTEGER.
C 21 Ime u naredbi EXTERNAL nalazi se također u naredbi COMMON ili DIMENSION.
C 22 IFIX ili FLOAT u naredbi EXTERNAL.
C 23 Neispravna realna konstanta.
C 24 Neispravna cjelobrojna konstanta.
C 25 Više od 15 formalnih argumenata ili duplicirani formalni argument u funkcijskoj naredbi.
C 26 Desna zagrada manjka u indeksnom izrazu.
C 27 Sintaktička pogreška u naredbi FORMAT.
C 28 Naredba FORMAT bez broja naredbe.
C 29 Specificirana širina polja veća od 145.
C 30 U naredbi FORMAT u specifikaciji E ili F je širina polja veća od 127, ili je broj decimala veći od 31, ili je broj decimala veći od širine polja.
C 31 Pogreška kod indeksa u naredbi EQUIVALENCE.
C 32 Indeksirana varijabla u funkcijskoj naredbi.
C 33 Neispravno formiran indeksni izraz.

C 34 Nedefinirana varijabla u indeksnom izrazu.

C 35 Količina izraza u indeksnom izrazu ne slaže se s naredbom gdje se definira dimenzija.

C 36 Neispravna aritmetička naredba ili varijabla, odnosno u potprogramu FUNCTION lijeva strana jedne aritmetičke naredbe je formalni argument (ili se nalazi u naredbi COMMON).

C 37 Sintaktička pogreška u naredbi IF.

C 38 Neispravni izraz u naredbi IF.

C 39 Sintaktička pogreška ili neispravni jednostavni argument u naredbi CALL.

C 40 Neispravni izraz u naredbi CALL.

C 41 Neispravni izraz lijevo od znaka jednakosti u funkcijskoj naredbi.

C 42 Neispravni izraz desno od znaka jednakosti u funkcijskoj naredbi.

C 43 U naredbi IF, GO TO ili DO broj naredbe nedostaje, neispravan je, krivo je smješten, ili je to broj naredbe FORMAT.

C 44 Sintaktička pogreška u naredbi READ ili WRITE.

C 45 Naredba IOCS nedostaje za odgovarajuću naredbu READ ili WRITE (samo za glavni program).

C 46 Broj naredbe FORMAT nedostaje, ili je neispravan, u naredbi READ ili WRITE.

C 47 Sintaktička pogreška u ulazno-izlaznoj listi, ili neispravan element u listi, odnosno u potprogramu FUNCTION ulazni element u listi je formalni argument ili se nalazi u naredbi COMMON.

C 48 Sintaktička pogreška u naredbi GO TO.

C 49 Indeks izračunate naredbe GO TO manjka, neispravan je, ili ispred njega nema zareza.

C 50 Postoje upravljačke naredbe TRANSFER TRACE ili ARITHMETIC TRACE, a nema upravljačke naredbe IOCS u glavnom programu.

C 51 Neispravno uklopljenja naredba DO, odnosno posljednja naredba u petlji DO je naredba GO TO, IF, RETURN, FORMAT, STOP, PAUSE ili DO.

C 52 Više nego 25 uklopljenih naredaba DO.

C 53 Sintaktička pogreška u naredbi DO.

C 54 Početna vrijednost u naredbi DO je nula.

C 55 U potprogramu FUNCTION indeks naredbe DO je formalni argument ili se nalazi u naredbi COMMON.

C 59 Sintaktička pogreška u naredbi STOP.

C 60 Sintaktička pogreška u naredbi PAUSE.

C 61 Cjelobrojna konstanta u naredbi STOP ili PAUSE je veća od 9999.

C 62 Posljednja izvršna naredba prije naredbe END nije STOP, GO TO, IF, CALL LINK, CALL EXIT ili RETURN.

C 63 Naredba sadrži više od 15 različitih indeksnih izraza.

C 64 Naredba preduga za obradu, bilo zbog kompilatorskog proširenja izraza za indekse, bilo zbog kompilatorskog dodavanja proizvedenih lokacija za privremeno memoriranje.

C 65 Sve varijable su nedefinirane u listi EQUIVALENCE (vidi primjedbu).

C 66 Varijabla načinjena ekvivalentnom nekom elementu polja na taj način, da izaziva protezanje polja preko početka područja COMMON (vidi primjedbu).

C 67 Dvije varijable ili dva elementa polja u naredbi COMMON, odnosno relativne lokacije dviju varijabli ili dvaju elemenata polja poistovećeni su više nego jedanput (direktno ili indirektno). (Vidi primjedbu.)

C 68 Sintaktička pogreška u naredbi EQUIVALENCE, ili nedopušteno ime varijable u listi EQUIVALENCE.

C 69 Potprogram ne sadrži naredbu RETURN, ili glavni program sadrži naredbu RETURN.

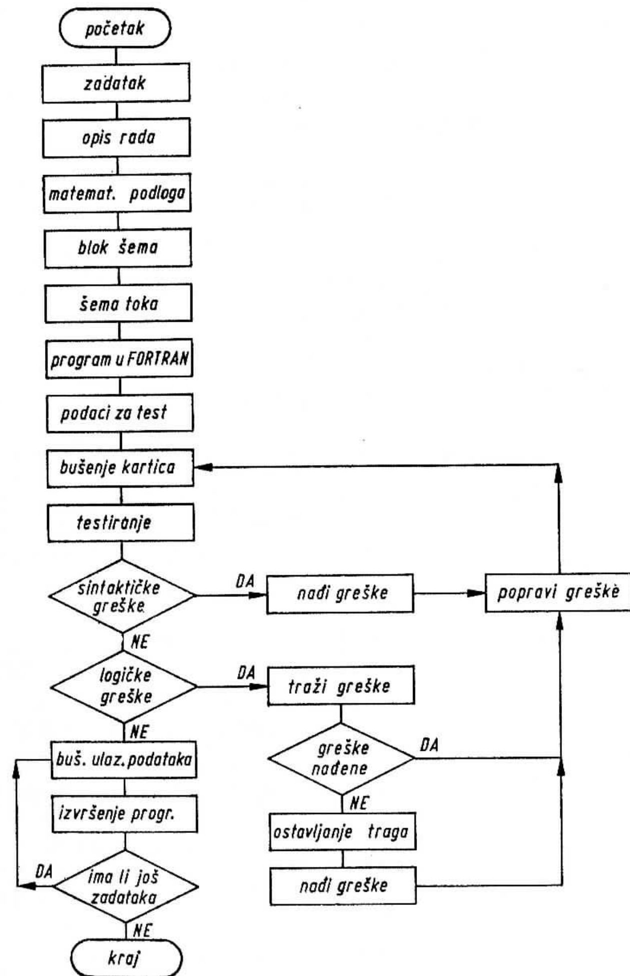
C 70 Nema naredbe DEFINE FILE u glavnom programu, koji sadrži naredbu READ, WRITE ili FIND za disk.

C 71 Sintaktička pogreška u naredbi DEFINE FILE.

C 72 Duplicirana naredba DEFINE FILE, više od 75 naredaba DEFINE FILE ili naredba DEFINE FILE u potprogramu.

C 73 Sintaktička pogreška u broju zapisa kod naredbe READ, WRITE ili FIND.

Primjedba: Otkrivanje jedne pogreške pod brojem 65, 66 ili 67 sprečava svako daljnje otkrivanje bilo koje od ove 3 pogreške.



sklapanje
III sklopke
ploča
svijetlo
centralna jedinica
1 riječ
1 riječ
čitalo kartica
kod
pamćenje računanje organiziranje
kod
kod
pisaljka
bušilo kartica
tastatura
kod
zapisna sprava
tiskaljka
1 bit
1 bit
tipke
signali
svijetlo
svijetlo
vanjski ulaz
alfanum
vanjski izlaz
sistem IBM 1130
bušeno
alfanuma
alfanuma
bušeno
graf
bušilo
bušilo
spremište kartica
priprema kartica
elektronička računala
alfanuma
alfanuma
graf
programeri
priprema programa
računski centar
problem
rješenje

DODATAK

Ono što je navedeno u ovom dodatku vrijedi za UNIVAC serije 1100 (za razliku od IBM 1130). Razlike su navedene uz onu lekciju, gdje se prvi put pojavljuju, ali — naravno — vrijede i za daljnje lekcije.

Uz predgovor

— Umjesto računala IBM 1130 s upravljačkim jezikom MONITOR, u dodatku se uzima u obzir računalo UNIVAC Serije 1100 s upravljačkim jezikom EXEC 8.

Uz 2. lekciju

— Karakteristika linijske tiskaljke je 6 (a ne 3).
— Linijska tiskaljka tiska 132 znaka u retku, brzinom do 1600 redaka na minutu.
— Osim slova engleske abecede mogu se upotrijebiti i slova: Ć, Č, Đ, Š, Ž.

Uz 4. lekciju

— Ime varijable može imati najviše 6 znakova (a ne najviše 5).

Uz 5. lekciju

— Najveća cjelobrojna konstanta je 34 359 738 367.
— Najmanja cjelobrojna konstanta je — 34 359 738 367.

Uz 6. lekciju

— Maksimalna vrijednost cjelobrojnog rezultata je 34 359 738 367.
— Minimalna vrijednost cjelobrojnog rezultata je — 34 359 378 367.

Uz 12. lekciju

— Produženje naredbe smije biti najviše 19 (a ne 5).

Uz 17. lekciju

— Navedene monitorske naredbe ne vrijede za UNIVAC.
— Upravljačke naredbe za EXEC 8 nisu navedne u ovom dodatku (to ionako ne spada u FORTRAN).

Uz 18. lekciju

— Pogreške u programu bit će drugačije navedene, kako to odgovara upravljačkom jeziku EXEC 7 (u što ovdje opet nećemo ulaziti).

Uz 19. lekciju

— Ime varijable može imati najviše 6 znakova (a ne 5).
— Realna veličina ima točnost od 9 znamenaka (a ne 7).

Uz 20. lekciju

— Čitalo kartica ima kao karakteristiku brojku 5 (a ne 2).

Uz 28. lekciju

— Broj uz STOP može biti najviše 999999.
— Nakon privremenog zaustavljanja, rad se nastavlja ako se pritisne na tipku S.

Uz 30. lekciju

— Monitorske naredbe ne vrijede za UNIVAC.

Uz 31. lekciju

— Količina indeksa može iznositi do sedam (a ne do tri).

Uz 32. lekciju

— Indeksni izraz može biti nešto složeniji (u to ovdje ne ćemo ulaziti).

Uz 36. lekciju

— Realnoj varijabli može se pridati 6 znakova.
— Cjelobrojnoj varijabli može se pridati 6 znakova.

Uz 37. lekciju

— Potprogram mora biti spremljen u računalu (ne mora biti isključivo na disku).

Uz 40. lekciju

— Prikazani način spremanja na disk ne odgovara za UNIVAC.

Uz 41. lekciju

— Prikazano pozivanje programa ne odgovara za UNIVAC.

Uz 42. lekciju

— Prikazane monitorske naredbe ne odgovaraju za UNIVAC.

Uz prilog 2

— Navedeni popis pogrešaka ne vrijedi za UNIVAC.
— UNIVAC serije 1100 daje obavijesti o pogreškama punim tekstom na engleskom jeziku (a ne putem šifre), pa njihov popis nije potrebno ovdje posebno navoditi.

KAZALO

Znak: 7510 Sv

Izdanje:

Prof. dr. ing. BOŽIDAR STEFANINI

FORTRAN — UDŽBENIK PROGRAMIRANJA

Izdavač:

TEHNIČKA KNJIGA, izdavačko poduzeće
OOUR IZDAVAČKA DJELATNOST
ZAGREB, Jurišićeva 10

Za izdavača

Glavni urednik:

Ing. ZVONIMIR VISTRIČKA

Urednik edicije:

Ing. IVAN UREMOVIĆ

Lektorirao:

ZVONIMIR VELJAČIĆ

Korigirao:

AUTOR

Naslovna strana:

MARIJAN OREŠIĆ

Tisak:

ŠTAMPARIJA »OBOD« CETINJE

Tisak dovršen:

U OŽUJKU 1975.

Oslobođeno osnovnog poreza na promet na temelju mišljenja Republičkog sekretarijata za prosvjetu, kulturu i fizičku kulturu SR Hrvatske.



TEHNIČKA KNJIGA
ZAGREB